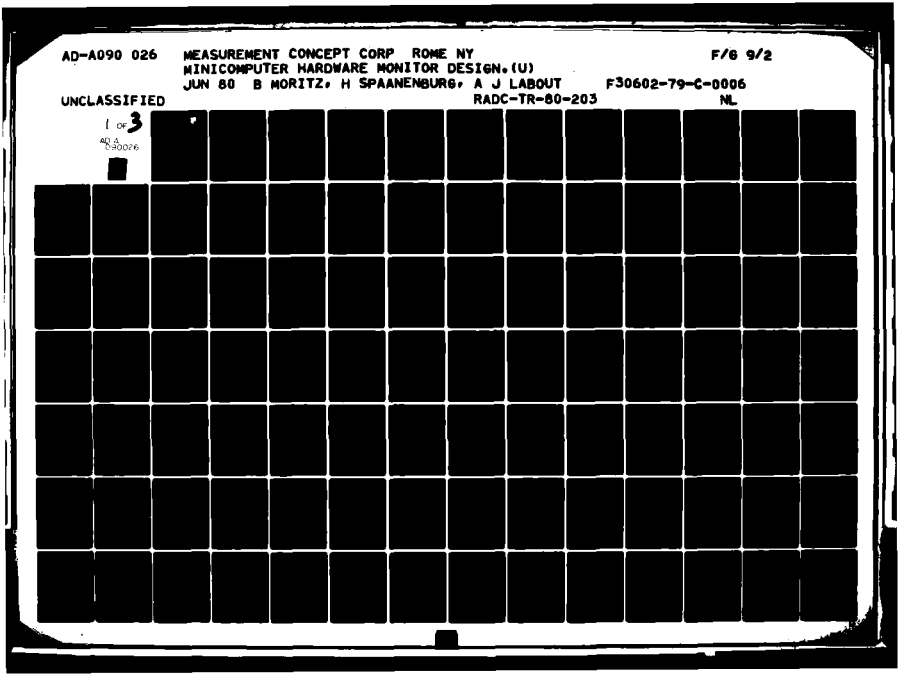
AD-A090 026 MEASUREMENT CONCEPT CORP ROME NY F/G 9/2
MINICOMPUTER HARDWARE MONITOR DESIGN.(U)
JUN 80 B MORITZ, H SPAANENBURG, A J LABOUT F30602-79-C-0006
UNCLASSIFIED RADC-TR-80-203 NL

1 OF 3
AD A 090026



AD A090026

LEVEL II

12

RADC-TR-80-203
Final Technical Report
June 1980

MINICOMPUTER HARDWARE MONITOR DESIGN

Measurement Concept Corporation

B. Moritz
H. Spaanenburg
A. J. Labout

DTIC ELECTE OCT 7 1980 C

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

DDC FILE COPY

80 10 7 071

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-80-203 has been reviewed and is approved for publication.

APPROVED:

JOHN E. FRANK
Project Engineer

APPROVED:

OWEN R. LAWTER, Colonel, USAF
Chief, Intelligence & Reconnaissance Division

FOR THE COMMANDER:

JOHN P. HUSS
Acting Chief, Plans Office

SUBJECT TO EXPORT CONTROL LAWS

This document contains information for manufacturing or using munitions of war. Export of the information contained herein, or release to foreign nationals within the United States, without first obtaining an export license, is a violation of the International Traffic in Arms Regulations. Such violation is subject to a penalty of up to 2 years imprisonment and a fine of $100.000 under 22 U.S.C 2778.

Include this notice with any reproduced portion of this document.

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRDA), Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-80-203	2. GOVT ACCESSION NO. AD-A090026
4. TITLE (and Subtitle) MINICOMPUTER HARDWARE MONITOR DESIGN	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report Nov 78 - Nov 79
	6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) B. Moritz H. Spaanenburg A. J. Labout	8. CONTRACT OR GRANT NUMBER(s) F30602-79-C-0006
9. PERFORMING ORGANIZATION NAME AND ADDRESS Measurement Concept Corporation 1721 Black River Boulevard Rome NY 13440	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45941606
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRDA) Griffiss AFB NY 13441	12. REPORT DATE June 1980 13. NUMBER OF PAGES 226
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same	15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: John E. Frank (IRDA)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Monitor(s)
Hardware Monitors
AN/GYQ-21(V)
Computer Performance Evaluation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report has documented the exploratory research that has been conducted for the development of an AN/GYQ-21(V) hardware monitor.

In Section 2.0 a description has been provided of the concept of monitoring, the users involved with monitoring and the monitoring systems that are already in existence. The objective was to put the development of the hybrid monitor into its proper perspective.

DD FORM 1473 EDITION OF 1 JAN 73 IS OBSOLETE

UNCLASSIFIED (Cont'd)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Item 20 (Cont'd)

Section 3.0 first of all examines the AN/GYQ-21(V) in detail, ~~then describes~~ its interfaces to the monitor. The measurement set considered to be a baseline for performance monitoring is documented in detail as well as the locations of its requisite probe-points. The major contribution towards the hardware monitor design can be found in the break-up of measurements into three stages: the vector generator, the programmable patchboard and the monitor microprocessor. The eventual hardware specification of those devices awaits the decisions that have to be made concerning the technology involved.

In Section 4.0 specifications for the software involved have been documented. Foremost in the designers' mind has been the ease of user interface with the monitor.

All of the technical and software/organizational techniques presented in the report are within the state-of-the-art. As a result, a microprocessor based performance monitor possessing an extraordinary degree of applicability to all potential users is eminently achievable.

User acceptance has been guaranteed by the lack of user-installed probes and the versatile user interface high level language software.

In short, it is believed that the simplicity of staged measurement handling, maximum utilization of nonsensitive probe interfaces such as the 21(V) busses, and the integration of passive hardware, active hardware, software, and cooperative hybrid measurement/control techniques provide RADC and the end user with a highly useful, low risk product at a cost commensurate with the complexity of the user requirement.

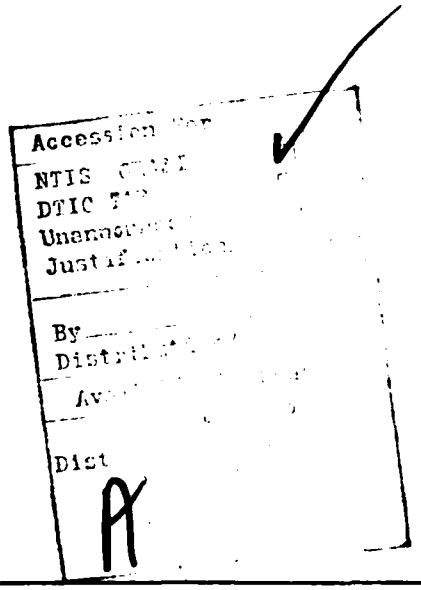


TABLE OF CONTENTS

		Page
1.0	INTRODUCTION	1-1
1.1	Measurement Concepts	1-1
1.2	Basic Operational Measurement	1-5
1.3	Categorization of Computer Systems Monitor . . .	1-6
1.4	Mc^2 Hybrid Monitor Design	1-9
1.5	Outline of Report	1-10
2.0	SYSTEM CONCEPT AND SCOPE	2-1
2.1	Monitoring Requirements	2-1
2.2	Monitor Survey	2-8
2.2.1	Software-Based Monitoring Systems	2-8
2.2.2	Hardware Monitoring Systems	2-20
2.2.3	Commercial Logic Analyzers	2-32
2.2.4	Hybrid Monitoring Concept	2-37
3.0	HARDWARE DESCRIPTION	3-1
3.1	AN/GYQ-21(V) System Description	3-6
3.1.1	UNIBUS	3-8
3.1.2	PDP-11/70 High Speed Bus	3-11
3.1.3	CPU/Console Interface	3-13
3.1.4	Multiprocessing AN/GYQ-21(V)s	3-16
3.1.5	AN/GYQ-21(V) Restrictions/Limitations	3-17
3.2	AN/GYQ-21(V) Monitor Probe Point Locations . . .	3-25
3.2.1	Interfaces Between AN/GYQ-21(V) and Hardware Monitor	3-28
3.2.2	Description of Interface Configuration	3-39
3.2.3	Measurement Set and Probe Points	3-43

TABLE OC CONTENTS (Continued)

		Page
3.3	Vector Generator	3-68
3.3.1	Vector Definition	3-69
3.3.2	Vector Building Components	3-72
3.3.3	Vector Post-Processing	3-75
3.4	Programmable Patchboard	3-76
3.4.1	Selection of Events	3-77
3.4.2	Counting of Events	3-79
3.4.3	Programmable Patchboard Realization	3-80
3.5	Monitor Control Unit (MCU)	3-82
3.5.1	Digital Equipment Corporation's LSI-11/23	3-84
3.5.2	Texas Instruments' TMS 9900	3-86
3.5.3	Bunker Ramo's IEP (Information Exchange Processor) Microprocessor	3-87
3.6	Unibus Interface Card	3-91
4.0	SOFTWARE FOR HYBRID MONITOR	4-1
4.1	Software Organization	4-3
4.2	User Interface	4-10
4.2.1	Event Definition Statements	4-13
4.2.2	Measurement Definition Statements	4-16
4.2.3	Additional Statistics Statement	4-17
4.2.4	Automatic Statistics	4-21
4.2.5	Module Manipulation Statements	4-22
4.2.6	Monitoring Options	4-24
4.2.7	Monitor Invocation	4-26
4.3	Host Software Monitor	4-28

TABLE OF CONTENTS (Continued)

		Page
4.3.1	Fundamental Software Monitor Techniques	4-31
4.3.2	Software Monitor (SOFMON) Measurement	4-35
4.3.3	Extended SOFMON Measurements	4-38
4.4	Measurement Collection Software	4-40
4.4.1	Bootstrap	4-42
4.4.2	Microprocessor Interrupt Servicing	4-43
4.4.3	Background Operations	4-46
4.5	Reporting Formats	4-47
4.5.1	Event Measurements	4-47
5.0	SUMMARY	5-1
6.0	REFERENCES	6-1

LIST OF FIGURES

Figure No.		Page
2-1	User Classes Hierarchy	2-7
2-2	Mc^2 Software Monitor	2-17
2-3	COMTEN Basic Monitor Design	2-30
2-4	TESDATA High-End Monitor Design	2-31
2-5	Wired-Program Monitor	2-39
2-6	Stored Program Configuration	2-41
2-7	A Hybrid Monitor	2-43
3-1	Monitor System Design	3-4
3-2	UNIBUS Priority Arbitration	3-10
3-3	PDP-11/70 vs. PDP-11/45	3-18
3-4	Probe Point Locations	3-26
3-5	Hardware Monitor Configuration (Boards)	3-40
3-6	Basic Vector Generator Components	3-73
3-7	Programmable Patchboard	3-81
3-8	DR11-B System	3-92
4-1	Create and Save Patch	4-5
4-2	Creating Linked Load Module to Initialize and Initiate Monitor.	4-6
4-3	Host Software Monitor (SOFMAN) Configuration	4-29
4-4	ESX EMT Handling By The Interceptive Monitor (SOFAMO)	4-33
4-5	Microprocessor Memory Layout	4-41
4-6	Transfer Command Block	4-44
4-7	Physical Unit Device Report	4-52
4-8	Automatic Statistics Report	4-54
4-9	Node Pool Summary Output	4-55

LIST OF FIGURES (Continued)

Figure No.		Page
4-10	System Performance Statistics Report	4-56
4-11	UNIBUS Statistics Report	4-58
4-12	Fast Bus Statistics Report	4-59
4-13	Mass Bus Statistics Report	4-60
4-14	Kiviat Graph of Optimum CPU-I/O Utilization	4-62
4-15	Instruction Type Report	4-64
4-16	Histogram of Disk Arm Movements for DPØ	4-66
4-17	Histogram of Counts for Each Cylinder Accessed on DPØ	4-67

LIST OF TABLES

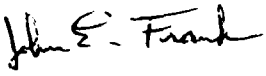
Table No.		Page
1-1	System Efficiency	1-3
1-2	System Effectiveness	1-4
3-1	Summary of Probe Points	3-27
3-2	UNIBUS Interface	3-29
3-3	UNIBUS Pin Assignments (By Signal Name)	3-30
3-4	PDP-11/45 Console/CPU Interface	3-33
3-5	PDP-11/70 Console/CPU Interface	3-34
3-6	Maintenance Interface	3-36
3-7	Discrete Points	3-37
3-8	Events and Vectors	3-71

EVALUATION

The U.S. Air Force has made a major commitment to the use of minicomputers as the vehicle for implementing many of its system requirements, a prime example being to support the sophisticated AN/GYQ-21(V) system. Whereas the individual minicomputer configuration is much cheaper than a large mainframe computer, many typical programs today involve multiple minicomputers and multiple sites with aggregate cost often equalling the cost of large mainframe systems. In the process of developing sophisticated systems using minicomputers it has become evident that equally sophisticated tools are required to optimize the performance of such systems. Since the use of hardware monitors has proven successful in optimizing the use of large mainframe systems, it is logical to introduce the use of hardware monitors into minicomputer systems. The microprocessor appears to be the ideal means to support a minicomputer hardware monitor. Economic benefits would likely equal, if not exceed, benefits large mainframe systems have derived through the use of hardware monitors. The introduction of a microprocessor-based hardware monitor into a minicomputer system would provide many operational benefits. The more restricted capacity of a typical minicomputer requires precise analysis of the timings of modules in the system as well as close monitoring of utilization of various hardware components. When a system becomes operational the hardware monitor can provide on-going (but not necessarily continuous) monitoring at each site to determine when specific machines are becoming overloaded and will aid in defining the most economical and most efficient methods of alleviating problems caused by growing workloads and diverse operational environments. These monitors will also prove highly valuable as an aid in system debugging. Minicomputers typically lack the types of diagnostic tools that are being developed for use with larger machines. Within reasonable economic constraints, such tools introduced into the minicomputer environment will greatly reduce the cost and time delay in debugging and checking out systems. The hardware monitor would have the added advantage of being operating-system independent and computer dependent, though each minicomputer in turn must be carefully analyzed to determine the appropriate probe points for collecting data. The performance of a PDP-11/70, for example, can be greatly affected by the degree to which work is performed through the cache memory. In order to know whether system performance should be enhanced by optimizing the code for increased use of cache, it is first necessary to have accurate readings for the percentage of executions currently being accomplished out of the cache memory. There are many other similar considerations. A hardware monitor will provide the technological vehicle to gather that information. The problems associated with conventional hardware monitors are that they are designed for use with large machines and they tend to be very expensive utilizing fairly large minicomputers

as integral components. What is required is a scaled-down system with economics sufficiently attractive so that it will be possible to replicate these monitors at a number of sites, and to make them readily available to all development projects. The basic system would be usable with any machine; the cost for developing and checking out the data collection pattern for a particular machine is a one-time expense.

This program determined the scope of monitoring capabilities that are economically feasible to implement using microcomputers to monitor minicomputers. Specifically addressed is the system monitoring of the AN/GYQ-21(V) computer system. The AN/GYQ-21(V) was analyzed and appropriate probe points were identified for collecting the data necessary for evaluating AN/GYQ-21(V) system performance and utilization.



JOHN FRANK
Project Engineer

1.0 INTRODUCTION

RADC has been a pioneer in the cost effective application of minicomputer technology to intelligence and C^3. The 21(V) has been the primary vehicle that carried these efforts to operational systems at USAFE, DIA, US Army, SAC, and other installations around the world. The diversity of applications has resulted in a multiplicity of hardware and software configurations that have, at times, been extremely difficult to optimize. To this is added existing and planned developmental programs that will both upgrade operational systems and create new ones.

The combined technical and management load of the 21(V) program has amply demonstrated the need for monitor and SPM support. The following sections present the technological basis from which this support may be obtained.

1.1 Measurement Concepts

The purpose of system performance evaluation is to determine how well a system satisfies (or will satisfy) the processing requirements of the system user. Evaluation takes place with respect to the objectives and goals of the organization served by the system. System performance may be graded on two aspects: system efficiency and system effectiveness. Efficiency signifies an evaluation of the internal activities of the system and of the utilization of individual system components. Effectiveness represents the evaluation of the capability of the system to process a given workload and to meet the response time requirements of the system users.

In order to evaluate a system from either of the above aspects, performance criteria (standards with which comparison can be made) must be established. Performance criteria can be specified only with respect to the type and the purpose of the subject system, its workload and the purpose of the evaluation. On the basis of the selected criteria, a set of evaluation aids may be chosen.

There are a number of evaluation aids utilized to measure system performance. Evaluation aids, if properly used, provide performance measurement statistics for use in determining both the efficiency and the effectiveness of a system. Tables 1-1 and 1-2, from Svobodova (1976), provide examples of system performance measures (evaluation aids) that have been of interest to computer system users.

System performance evaluation (a superset of system performance measures) is based upon the measurements (and the measures and statistics derived therefrom) that are collected during system operational periods (the data). These data may then be analyzed and reports can be produced apart from the system being studied.

The nature of the data collected is dependent upon the nature of the system being evaluated and the availability of appropriate "sensors". One of the most simple (and least expensive) of sensors is the common stopwatch. Although woefully inadequate for measuring microsecond and millisecond intervals, the stopwatch has a key role in determining efficiency measures such as CTMF and effectiveness measures such as turnaround time. This common stopwatch, when electronically coupled to internal system events, may still be adequate to determine internal delay factors, gain factors, system availability and relative throughput to sufficient accuracy.

For many other measures, the stopwatch and its more advanced brethren are supplemented with more sophisticated sensors such as hardware and software monitors and job logging (accounting) routines. In many cases where the necessary sensors are not available, the analyst will resort to simulation and modeling to represent the perceivable (measureable) system characteristics in terms of more basic operational efficiency factors. In these cases both measurement and modeling could be required in order to perform the necessary evaluation.

Performance Measure	Description
External delay factor	Ratio of job turnaround time to job processing time
Elapsed time multiprogramming factor (ETMF)	Ratio of job turnaround time under multiprogramming to job turnaround time when only one job is in the system
Gain factor	Ratio of total system time* needed to execute a set of job under multiprogramming to the total system time needed to execute the same set sequentially
CPU productivity	Percent of time a CPU is doing useful work (used as a measure of throughput)
Component overlap	Percent of time two or more system components operate simultaneously
System utility	Weighted sum of utilization of system resources
Overhead	Percent of CPU time required by the operating system
Internal delay factor	Ratio of the job processing time under multiprogramming to the job processing time when it is the only job in the system
Reaction time	Time between entering the last character on a terminal or receiving the input in the system and receiving first CPU quantum
Wait time for I/O	Elapsed time required to process an I/O task
Wait time for CPU	Elapsed time required to process a CPU task
Page fault frequency	Number of page faults per unit of time

Table 1-1 SYSTEM EFFICIENCY

* System time is the time during which at least one of the system's processors (CPU, channels) is busy.

Performance Measure	Description
Throughput	Amount of useful work completed per unit of time with given workload
Relative throughput	Ratio of the elapsed time required to process a given workload on system S_1 to the elapsed time required to process the same workload on system S_2
Capability (capacity)	Maximum amount of useful work that can be performed per unit of time with given workload
Turnaround time	Elapsed time between submitting a job* to a system and receiving the output
Response time	Turnaround time of requests and transactions in an interactive or real time system
Availability	Percentage of time a system is available to users including MTBF/MTTR statistics

Table 1-2 SYSTEM EFFECTIVENESS

* In an interactive system, each command issued from a user's terminal is assumed to represent a new job.

1.2 Basic Operational Measurements

Although many of the measures presented in Tables 1-1 and 1-2 are virtually identical to a basic physical measurement (e.g., time), many measures are derived from statistical and logical operations performed upon a set of basic measurements. The selected set of basic measurements also depends upon the evaluation criteria. However, they may be classified to include the following basic measurement types:

- o The occurrence of a hardware event
  - Interrupts, interrupt requests, and interrupt grants
  - I/O channel data transfer
  - Change of machine status
  - Execution of a specific instruction
  - Access of a specific memory location (or range)
- o The occurrence of a sequence of hardware events
- o The occurrence of a software event
  - Starting/stopping execution of a specific job
  - Request for a specific executive service
  - Calling of a specific subroutine
- o Hardware status or software status
  - Useful for "Gating" of events
  - ANDing and ORing of several state variables
  - CPU, I/O channels, and peripherals
- o Hardware Timing
  - Interval between any two hardware events
- o Software timing

- o The counting of events
  - Hardware or software
  - Single events or event sequences
- o Event tracing
  - The recording of events and the time of their occurrence when multiple event types may be involved

In short, the basic measurements consist of event detection, event counting, tracing, status, and interval timing. From these basic measurements come distributions of the values for these measurements, statistical measures of those distributions, and other mathematical measures obtained from the statistical measures and basic measurements (ratios, sums, etc.). The set of measures (the evaluation aids) contribute to the performance evaluation.

1.3 Categorization of Computer Systems Monitors

Performance monitoring of a computer system can be implemented along many basically different methodologies - each methodology with its own characteristics strengths and weaknesses. The following five classic monitor types have been built and/or designed:

- o Software monitor, a collection of programs on the measured machine that interrupt the execution of the system at the occurrence of specific system events. The software monitor then collects desired information before relinquishing control back to "normal" operations.
- o Firmware monitor, a monitor that is imbedded within the regular microcode instruction interpreter of a microprogrammable machine. This technique can minimize the interference (artifacts) that a slower software monitor introduces and is also capable of collecting detailed information unavailable to a software monitor.

- Logic Analyzer, a measurement device at the circuit level, monitoring and displaying logical signals.
- Hardware monitor, a free-standing imdependent physical device connected to the measured machine by cables and probes. In principle it does not disturb the operation of the measured machine.
- Hybrid monitor, a combination of hardware, firmware and software monitor.

Two primary monitor data collection methods are common:

- Event driven systems in which the monitor is triggered by specific significant events. These events could be job oriented or computer component oriented.
- Sampling systems in which the monitor contains an interval timer that will trigger data collection at certain time intervals.

Event driven systems are more efficient if specific events are to be measured. Sampling systems tend to generate much more data but they are useful in determining gross operational statistics (utilization rate, etc.).

Given a technique for accessing the events to be measured in the target machine, the detail of data available to the analyst and the time at which the data becomes available are important elements in performing the desired evaluation. Two methods of data recording are prevalent - traces and summaries.

- Tracing monitors produce the complete, time ordered, event by event listing of measured activities.
- Summary monitors produce a condensed version of the history of certain activities (i.e., a summary of activity over a given interval) in the form of a count, histogram, etc. with little or no precision in time ordering of events over short intervals.

Tracing is not commonly used in system-wide evaluations because of the extremely large amounts of information that must be stored, retrieved and reviewed; it is, however, useful for program debugging. Summary data monitors do allow the analyst to evaluate variable performance over longer periods through the "snapshot" mechanism - where the summary data is stored at regular intervals. Summary monitors, however, can record high speed event sequences with sequence detection logic at their front ends.

Most detailed evaluations require significant analysis of data obtained during a test. However, some critical data could be immediately displayed; this leads to two forms of monitor data presentation.

- o Real-time monitors where all desired information is immediately obtained and displayed or otherwise used in the measurement task. Only a limited amount of data can be meaningfully presented in this manner.
- o Post-processing monitors where the previously stored measurement data is retrieved from storage devices; it is processed, analyzed and reported at a later time then the actual measurement took place.

In Section 2.0 is described the principles involved in software, hardware and hybrid monitors. In each of them a distinction can be made between the following monitor components:

- o Instrumentation, such as software interceptive techniques, electronic probes, interfaces, etc.
- o Selector element, such as special hardware, vector generator, logical patchboards.
- o Processing element, such as host software, microprocessor monitor.

- o Recording element, such as host's main or secondary memory, counters, RAMs and any monitor storage devices.

- o Reporting element, such as host, monitor-microcomputer.

1.4 Mc^2 Hybrid Monitor Design

The object of this report is the documentation of exploratory research performed by Mc^2. This research has led to the development of design specifications for a minicomputer hardware monitor.

The Mc^2 design incorporates pure hardware, software, and hybrid elements. All of these types of elements are required for a completely synchronized, fully capable performance measurement program that generates minimal artifacts.

The combined hardware and software hybrid monitor can be used in hardware mode only (no host resident software) to capture basic hardware component performance data such as those popularly utilized and displayed in Kiviat and Gantt chart form (CPU busy, etc.); it can also be utilized together with a host software monitor or other cooperative code to provide powerful debugging tools and operating system software/job/program performance measures. Through the flexibility provided with programmable patchboard logic and the ability to easily select desirable functions to be active, the hybrid system approach will satisfy all currently known measurement and operational requirements at a low replication cost.

Independence from the target host operating system is assured by the hardware subsystem of the proposed monitor. For many job/program/overhead measurements, however, independence of host software is fictional as the requisite host software monitor element is specific to the host's operating system.

1.5 Outline of Report

In Section 1.0 an introduction was provided to the kind of measurements that generally are of interest to monitor service users. In addition the various forms of monitor services have been categorized.

In Section 2.0 further details of the monitor services categories have been provided, in order to establish a baseline of performance characteristics that should be included in the hybrid monitor design. Also described is the environment in which the AN/GYQ-21(V) monitor will be applied and the potential users that it will find in those environments.

Section 3.0 outlines the various hardware elements that are parts of the monitoring configuration. Included in this outline is a description of the AN/GYQ-21(V) environment, as it will be seen by the Monitor-21(V) interface. The measurements that can be made by the monitor, and that were of interest to the various users groups, are also described in Section 3.0. The description documents the contributions that the various hardware elements make to the measurements.

The software elements to be running on the host processor and monitor microprocessor have been specified in Section 4.0. The eventual implementation of the software elements is, of course, dependent on the ultimate decisions taken on hardware element implementations (choice of microprocessor, etc.).

Lastly in Section 5.0 a summary is provided of the cost factors involved in the implementation of the design, as well as its expected performance characteristics.

2.0 SYSTEM CONCEPT AND SCOPE

2.1 Monitoring Requirements

Commercial monitoring systems are oriented towards the user who is controlling a large data processing installation. Typically, the environment is a multiprocessor environment, with a flexible multiprogramming executive that must control a large number of users, a variety of processing requirements, large amounts of direct access storage and a mix of "batch" and on-line users. In about all cases, these environments are either IBM or IBM compatible (i.e., Amdahl) and the operating systems are almost always IBM products (i.e., OS-MVT, Multiprogramming, variable number of tasks) or the sophisticated virtual-storage operating system (VS2). Facility managers are tasked with the problems of "tuning" their facilities to provide the best possible workload throughput and to break dynamic workload "logjams" that may occur due to special transient conditions. Commercial monitors are designed to capture and process internal system information to assist management in identifying processing bottlenecks, and to forecast future processing requirements. The information is interpreted by persons having detailed knowledge of both the total system hardware configuration and the operating system or systems that run on the equipment under study.

The resultant information then influences management planning to accomplish performance enhancements through:

- Improved scheduling
  - batch and interactive jobs
  - development and testing
  - critical scheduled jobs
  - demand loading variations during a day

- o Optimization of system software runtime variables
- o Optimization of application programs
  - improved algorithms
  - reorganization of modules/overlays
- o Hardware reconfiguration
  - Channel/controller utilization
  - Multiprocessor peripheral sharing
- o Improvement of Data File organization
  - physical location of data
  - data structures
  - record contents

The environment of the AN/GYQ-21(V) certainly incorporates the above characteristics. It is expected that the "21(V)" facility managers will have basically the same operational enhancement goals as their commercial counterpart. However they are also burdened by additional factors that preclude many of the solutions available to the commercial facility manager:

- o In an intelligence/communications environment, the system must typically be operational on a 24 hour per day, 7 day per week schedule and management cannot predict a low level, crisis-free time for loading for testing during the midnight hours.

- The security arrangement does not stop with control of physical access or even with controlled remote access to data. In many cases the very presence of emanations - whether or not they contain comprehensible information - is compromising to the mission.

- The personnel has not typically grown with and learned the system over a long period of time. The military environment produces a constant change in personnel with the concomittent demands for continuous training and low complexity operations. For the same reason, one similarly cannot always rely on the availability of in-house expertise at the technical levels required for more esoteric analysis.

Solutions to this dilemma have not come easily or inexpensively and have typically involved acquisition of separate systems upon which software development and testing is performed. This situation does not necessarily aid in the correction of an intermittent system failure under "real" loads nor does it tend to make accurate loading and utilization data from the "live" system readily available so that well-directed, cost-effective improvements may be rapidly implemented.

Recognizing the situation and faced with the continuing placement of AN/GYQ-21(V) hardware and software at hundreds of critical locations around the world, RADC has taken several steps to provide the tools and techniques necessary to cost effectively optimize 21(V) performance and utility in its unique operational environment. The project, of which this document is the final report, is one of those steps.

The monitor described in this report is not solely of use for management. In general, the intended potential users of the AN/GYQ-21(V) monitor services can be partitioned into four "classes". Such partitioning, however, does not exclude, for instance, a "Class III" user from requiring "Class I" measurements. It does partition requirements into groups most frequently used by a "class" of monitor users.

- Class I Users

  Resident system technicians, as opposed to engineers, are the primary Class I representatives. Such technicians are responsible for the continued functioning of an AN/GYQ-21(V) system, primarily from a hardware standpoint. Responsible for some limited preventative and remedial maintenance, the technician would use the services supplied by the hybrid monitor to both verify proper continued system operation and to isolate system malfunctions. Such isolation would help the technician justify the use of outside or vendor-supplied maintenance, and would also help the technician to discuss the specific malfunction, in advance of their arrival, with the outside repair personnel. This will not only save valuable elapsed-time maintenance charge hours, but may also assist the outside maintenance personnel to assemble needed replacement components before making a charged visit to the facility where the defective AN/GYQ-21(V) is located.

- Class II Users

  Class II Users are software development people, primarily programmers. Their uses of the hybrid monitor would be for debugging and performance monitoring of isolated software modules, and are not as general in scope as any of the other user classes. Monitor facilities could be used to greatly enhance the functions of ODT (Online Debugging Technique), supplied with RSX operating systems. In addition, the hybrid monitor could provide the Class II user with a more sophisticated timing and resource usage analysis capability in a multi program/multi task environment.

- Class III Users

  Class III users are persons with responsibility for the continuing technical performance of a complete 21(V) system. It is this user that must answer to both Class IV (management) users, and indirectly to all system users, for the level of performance provided by the system. The Class III user is the most demanding user of hybrid monitoring services; they require access to information describing hardware operation in the most minute of detail, information about the performance of the Operating System, identification and parametric information regarding the transient user loads placed upon the system, and gate-level timing analysis information to resolve system deadlock conditions. While nearly all requirements of Class II and IV users could be gathered by means of host-supported software-based monitoring techniques, the Class III user requires information that can only be obtained by means of logic ancillary to the system being measured. The user's requirements most closely parallel the capabilities provided by a fast, logic state analyzer, with its list of required probe points being most demanding.

- Class IV Users

  Class IV users are management personnel responsible both for providing sufficient computer resources for a given "mission", and for justifying the cost of both present and future computer resources. They conform closely to the typical users of commercial software-based monitoring systems that report breakdowns of system utilization by user, time of day, etc. In addition, Class IV users need to be able to relate current demands upon a system to the system's maximal capacity in order to forecast system capability expansion. Most Class IV measurement requirements are cumulative, and do not require extensive real-time reporting capability.

In Figure 2-1, the relationship between the user classes is illustrated. In addition, the figure shows the way in which the various user classes complement each other.

In the following section, we will survey existing monitoring facilities and document attributes of these systems. The concepts thus described have been incorporated in the baseline concepts of Mc^2's minicomputer hardware monitor design, the objective of it being a scaling down of monitoring facilities from large-scale machinery towards minicomputer systems.

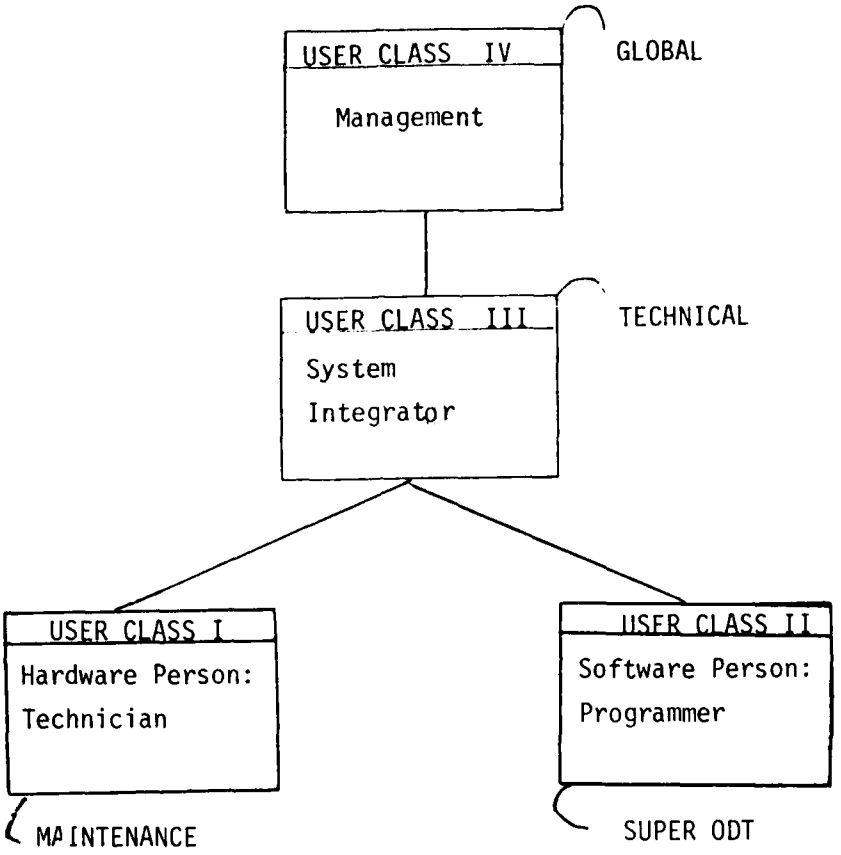


Figure 2-1 User Classes Hierarchy

2.2 Monitor Survey

2.2.1 Software-Based Monitoring Systems

Software-based monitoring methods utilize the resources of the system being measured to monitor its own activities. As every request to use one of the system's hardware resources is actually a computer instruction, or a sequence thereof, it is a relatively simple matter to divert all instructions in the midst of their normal fetch-execute cycle to a software routine that examines the instruction and determines what sort of resources it requires. The software routine used to decode the diverted "workload" instructions, is itself composed of instructions. It can easily be seen that even this elementary software based monitoring scheme introduces confusion into the collected results. While this "contamination", called artifact, can be accounted for and the results adjusted accordingly, it can be seen that the scheme introduces operational overhead to the system being measured.

The level of overhead introduced varies, in proportion, to the degree of detail of the information being collected. While it is true that in a properly-designed and properly-working system that nearly any collection of system functions can be quantified and recorded, the overhead of doing so on a continuous basis burdens the system being measured to an unacceptable level. While all software monitors must introduce overhead, the difference between the good ones and bad ones is more or less a function of their affect on the system's normal assigned operations. There are several clever compromises that a software-based measurement scheme can employ while not seriously compromising the integrity of the measurement being made. These compromises are the indirect marketing tools used by the handful of companies that produce software monitors for general commercial use. More will be said about both the compromises and the vendors in section 2.2.1.3.1.

Software monitors contain two major functions:

- o collection of performance measurements and/or their statistics during operation of the subject system.
- o analysis and reporting of the collected statistics, usually upon the termination of subject system processing.

2.2.1.1 Collection of Performance Measurements

The collection function is embedded within the subject system at the operating system or executive level in order to gain access to system management data relative to memory utilization, scheduling queues, peripheral device activity queues, CPU utilization, the system clock, etc. The collection function of a software monitor should operate as a near invisible component of the operating system. It should be noted that, for this reason, the monitor itself contributes to the measurement data being collected since CPU time, memory utilization and possibly I/O activity statistics will include monitor processing and storage areas. In the normal case, a software monitor will contribute from 2 - 5% of the measured quantities in the above statistics.

Two implementation methods are employed for the collection function in software monitors.

- o event driven
- o periodic sampling

An event-driven monitor is activated by the occurrence of a special event within the subject system that is able to trigger the monitor (e.g., an interrupt). The monitor examines the cause of the event and collects pertinent statistics dependent upon the type of event and/or the subject system function responsible for the event. Control is then transferred to

the normal operating system event/interrupt handling routines. An interrupt driven monitor obtains access to interrupts through modification of the subject operating system's interrupt transfer vectors whereby the transfer is directed to the software monitor, not to the operating system's interrupt routines.

Periodic sampling involves the scheduling of the monitor collection function at specified time intervals. When activated, the monitor examines all pertinent system tables and collects performance statistics reflecting the status of the subject system at that point in time.

2.2.1.2 Analysis and Reporting of Collected Statistics

The collected performance measurement statistics are analyzed and reported by a distinct monitor function which is independently activated at the user's convenience. The collected statistics may be reported in detail, summarized, or printed in graphic form (e.g., Kiviat or Gantt representations) - depending upon the nature of the subject system and upon the type of measurement data involved. The collected statistics may be saved in a peripheral file if the collection function has not been designed to do so. The reporting function of a software monitor may also be employed to perform comparison analysis of performance measurements from several operational periods of the subject system.

2.2.1.3 Types of Monitoring Supported by Commercial Software Monitors

There are three distinct types of software-based monitors. These can be differentiated on the basis of their measurement objectives as follows:

TYPE

1 Monitors oriented toward the collection of information about hardware component usage.

2 Monitors oriented towards the measurement of usage of executive resources.

3 Monitors oriented towards the measurement of system user throughput.

While the first kind of software monitoring is sometimes necessary to locate chronic bottlenecks, most commercial software monitors are of type 2 and/or 3. As such, these monitors are totally designed around a single operating system.

In Mc^2's survey of the software monitor market, the following conclusions were drawn:

- All were designed for "IBM" hardware/software environments. They would, however, support equivalent architectures, such as AMDAHL and ITEL/National Semiconductor.
- Each was designed for a specific operating system. MVS was most common, with CICS a close second.
- Except for Mc^2 monitor, no software monitor products have been made available for either the PDP 11/45 or 11/70.

The implication from these conclusions is that the software component of the hardware/software monitoring system designed by Mc^2 will also not be universal, but rather will be written specifically for one or more of the possible operating systems for the AN/GYQ-21(V); i.e., IAS, RSX-11D, RSX-11M (+), or perhaps UNIX. Further, as the hardware component of the hardware/software monitoring system will be better suited to the

measurement of hardware component usage, any software monitoring services will be type 2 or 3, although they will make use of the data collected by the hardware monitor to assess type 1 activity.

2.2.1.3.1 Features of Commercial Monitors

Two interesting, if not profound, ideas gained from examination of some of the more MVS-oriented monitors were monitor modularity (a Boole and Babbage concept) and the universally used monitor configuration.

The modular monitor was developed by B&B, as a "compromise" on the monitor's appetite for system resources. Basically, it means that a monitor "nucleus" exists, containing monitor clock, I/O and reporting routines, and its own "sub-executive". This nucleus is sort of the Venn-diagram overlap area of the variety of sub-monitors (types 1,2 and 3) that B&B provides. Dormant, the nucleus takes very little space, and very little system overhead. The monitor user can communicate with this nucleus, and cause it to call into memory one or more sub-monitor modules. The sub-monitor activated with input from the user reports about recording frequency, etc. The result of this modularity is that monitor overhead is reduced to just that necessary for the monitoring features needed at the time.

The "interactive monitor configuration" is usually set up by means of a high level English-like language that allows the user to communicate with the software monitor and tell it what the user wants collected, reported, how often and where.

An efficient software monitor is designed by determining the minimum frequency at which each important system component is used, and by forcing the monitoring process to be event driven. That is, to have a given resource sampled only when the monitor is informed that a significant event has occurred.

Unfortunately, the optimal rates for sampling system resources are as dynamic as the usage of system resources themselves. Further, since it is necessary to monitor the systems for a variety of distinct reasons (i.e., long-term management planning, individual performance problems), the monitoring "formula" for sampling and report rates will change as well. Most commercial monitors allow the user to adjust sampling of each important resource by direct interaction with the software monitor.

Principally different software monitors have been written by Boole and Babbage, Inc., !CANDLE Corp., and Mc^2. The Mc^2 monitor will be described in more detail in section 2.2.1.3.2 in order to illustrate the software monitor concept.

The surveyed software monitors can be characterized as follows.

Name: CONTROL/CICS (Customer Information Control System)

Vendor: Boole and Babbage

Hardware: IBM 370 series

Software: IBM CICS (Customer Information Control System)

Monitor Type: 3

Description: Provides management-oriented reporting on the performance of a CICS HW/SW system, with emphasis on system throughput.

Name: CONTROL/CMF (Comprehensive Management Facility)

Vendor: Boole and Babbage

Hardware: IBM 370 series

Software: IBM MVS

Monitor Type: 1, 2 and 3

Description: CMF (Comprehensive Management Facility) provides a resident "Master Monitor" onto which may be hung as many special Boole and Babbage supplemental monitors as are necessary Only monitors that are "active" consume system resources, and the total monitor configuration is under real-time control of a user. CMF can monitor from the hardware component level, through use of executive services and up to the system throughput level. It will support the generation of long term trending data bases as well as short-term high resolution data bases and special-purpose submonitor data bases.

Name: Omegamon

Vendor: !CANDLE CORP.

Hardware: IBM 370 series

Software: IBM MVS ·

Monitor Type: Some 1, mostly 2 and 3

Description: Primarily a real-time reporting system supporting IBM 3270 terminals in dedicated or remote (through TSO) mode. Primarily oriented to machine room management, Omegamon allows dynamic monitoring of system bottlenecks, "deadly embrace" and throughput delay, and help personnel remove bottlenecks or optimize their MVS system.

Name: Mc^2 AN/GYQ-21(V) software monitor

Vendor: Measurement Concept Corporation

Hardware: DEC PDP 11/45-compatible machines

Software: DEC RSX-11D Real-Time Executive

Monitor Type: Some 1, mostly 2

Description: The Mc^2 monitor was designed not only to monitor the behavior of an AN/GYQ-21(V) system, but also to examine the behavior of the Bunker Ramo Corporation SARP (Storage and Retrieval Processor) system.

2.2.1.3.2 The Mc^2 Software Monitor

A software monitor has been developed by Mc^2 to collect performance statistics of an AN/GYQ-21(V) computer. The present monitor collects and reports a limited set of performance measurements involving input/output activities. The monitor has been utilized to obtain I/O performance measurements for SARP* functions within the CATIS** system, which subsequently have been successfully used in a computer simulation model of that particular system.

The Mc^2 monitor has three main functions (Figure 2-2).

- o Definition of parameters
- o Collection of performance measurement data
- o Reporting of collected measurement data

The Parameter Definition Function allows the user to specify the measurement statistics to be collected during the operational period of the subject system. Parameters are defined before beginning the operational period of the subject system. A secondary purpose of the Parameter Definition Function is to initialize the runtime collection function of the monitor. The initialization process involves the allocation of main store storage areas for the collected statistics and the alteration of the internal interrupt transfer vectors within the RSX operating system to direct interrupt processing to the active monitor.

*SARP - Storage And Retrieval Processor - a Data Management System

**CATIS - Computer Aided Tactical Information System

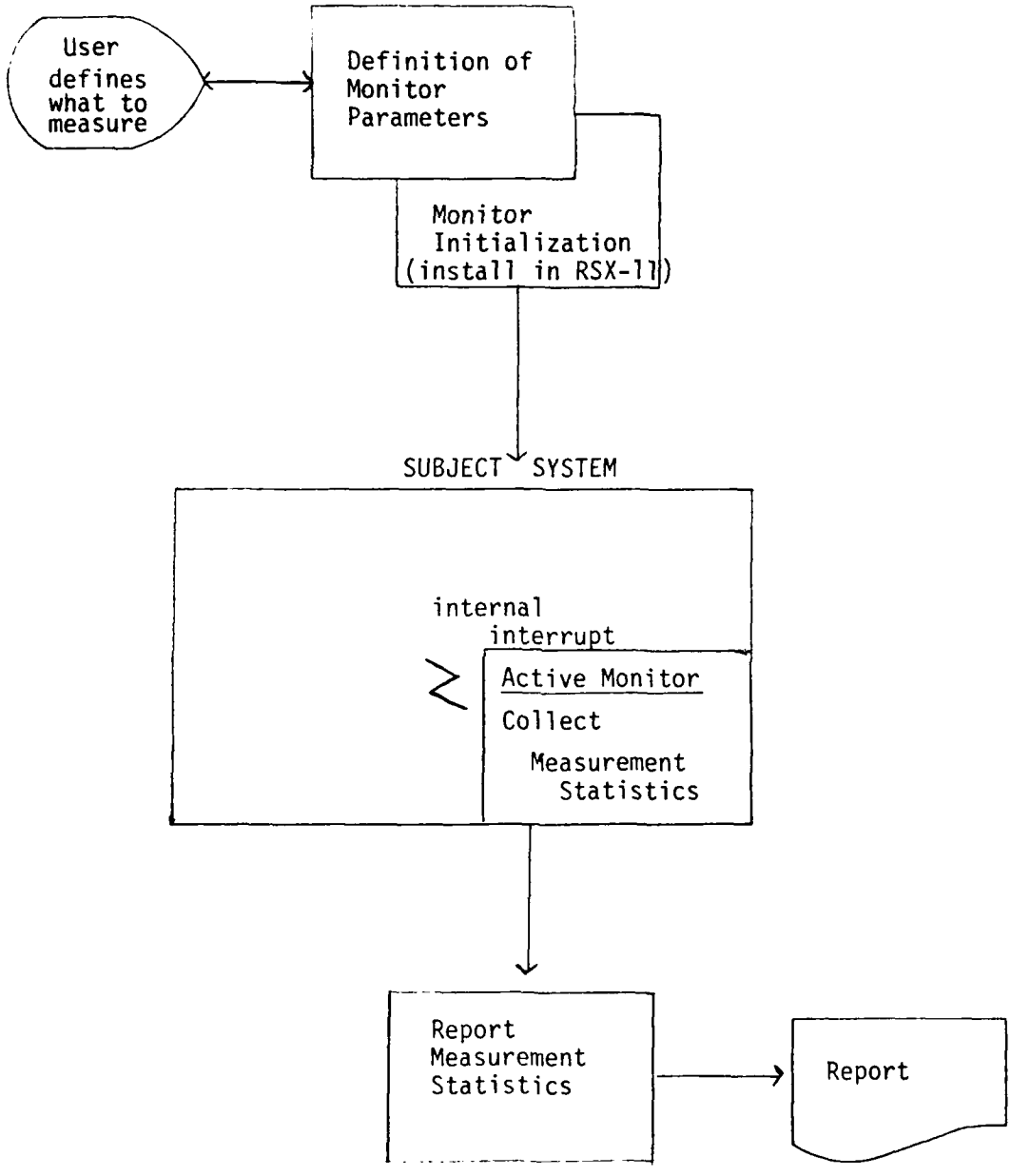


Figure 2-2 Mc^2 Software Monitor

Performance measurement data is collected while the subject system is in operation. The active monitor module captures all internally generated software interrupt requests for operating system services (an EMT), examines the request with respect to the user-defined collection parameters, and gathers the measurement statistics specified if the interrupt request is determined to be pertinent. The intercepted request is then passed to the RSX operating system to perform the requested service. All measurement statistics requested are accumulated in main store buffer areas.

The reporting function of the monitor is activated at the discretion of the user. The reporting function may be called during operational periods of the subject system or upon termination of operations by the subject system. The reporting function produces a report containing the performance measurements requested in the Parameter Definition Function.

Among the key performance measurements made by the Mc^2 monitor are:

- Count of RSX-11D I/O directives by task and (work in progress) by device
- Amount of time taken to service RSX-11D directives (by directive)
- Amount of CPU time utilized by task
- Disk activity measures
  - Arm motion histogram
  - Sector address I/O activity (also a histogram)
- Communications Device Activity Measure (work in progress)
  - Size of message transferred
- RSX-11D SYSCOM Node pool utilization

Summary measures (event counts and times) are collected and may be periodically dumped to disk for later "snapshot" time analysis.

The Mc^2 monitor is designed to operate as an integral part of the RSX-11D, version 6B Operating System for the AN/GYQ-21(V) computer. Nomenclature, mnemonics, absolute memory addresses and operating system table structure references used by the monitor are for the named version of RSX-11D. Buffer space allocation and usage take advantage of the RSX Task philosophy. The Mc^2 monitor is written in Macro-11.

All of the above contribute to the non-portability of the Mc^2 monitor. This restriction is common in software monitors which, in general, are designed for use with a specific CPU and operating system combination.

2.2.1.4 Firmware and Microprogramming

With the growth of microprogrammable mainframes, another version of the software monitors has emerged. In principle the microcode that interprets the machine's assembly (macro) level instruction is slightly modified to record (or to have recorded) appropriate performance data. When only the system's own resources are utilized, this technique becomes a potentially sophisticated software monitor with the capabilities to perform some hardware oriented measurements. It also suffers from the general drawback of software monitors: it introduces measureable artifacts.

2.2.2 Hardware Monitoring Systems

Measuring the performance of a computer system dedicated to the execution of a single task is a relatively easy undertaking. Multiprogramming presents additional obstacles to system performance measurement. Since processes of all active programs are interleaved in time, it is not easy to identify delays in processing due to contention between tasks for hardware and/or software resources. More information is necessary about the states of the CPU, I/O busses, peripheral controllers, and about memory utilization relative to time.

Capturing and recording the state and utilization of the various internal subsystems of a multiprogrammed data processing system can be performed in two ways: the system can be used to interrogate itself, or special hardware can be used to interrogate the various subsystems while the system runs its normal workload. These two approaches correspond to "software" and "hardware" monitoring, respectively.

Using a system to monitor itself is attractive from the standpoint of not having to acquire additional resources, but it imposes some restrictions upon the system it is measuring. By placing a complex software subsystem between the normal operating jobstream and the hardware resources necessary to service the jobstream, overhead is incurred, slowing down the system. Additionally, the monitoring process can itself create contention for services that would not exist if the monitoring were not being undertaken. Because of these problems, software monitoring is rarely used on a continuous basis to determine overall system performance.

Hardware monitoring, on the other hand, introduces no loading of the system being studied. Instead, carefully selected "probe" devices eavesdrop on the activities of various subcomponents of the system. As these probes are invisible to the system under measurement, an accurate accounting of the utilization of all hardware components of the system can be made without fear of the measurement process itself affecting these measurements. The sorts of data collected by hardware monitoring procedures include:

- Counting the occurrence of an event
- Measuring the duration of the event
- Determining the "state" of a device (i.e., active, wait, idle)
- Intercepting and storing data
- Determining memory segment utilization
- Registering the time at which the event occurred

The problem of monitoring the activity of a multi-programmed system, requires the monitor results to be linked with each particular task. Hardware monitoring procedures are often combined with some software monitoring to specifically equate hardware monitor statistics with a task.

While many manufacturers provide hardware designed to monitor specific areas of a system's performance, we will focus on two vendors who manufacture hardware based monitoring systems. The two vendors, COMTEN (acquired by NCR) and TESDATA provide integrated hardware, software, installation and customer training to help data processing management deal with the kinds of problems found in large computer installations. Other vendors supply monitoring equipment that classifies more accurately as hardware diagnostic test equipment.

COMTEN can be considered the forerunner in the hardware monitoring business, having filed for their patent in 1965 (granted in 1968), which dates their concern to the approximate timeframe of IBM's System 360. Their systems can be classified as "passive", in that they collect data for later processing by COMTEN software running on the system being monitored.

TESDATA markets a line of more sophisticated monitors, each based on a minicomputer, which controls the monitoring processes, reduces the resultant data, and reports the findings (both in real time and by batch process) to the user.

Each of these systems has elements in common with the other. The methods by which these monitors collect, record and report monitoring data is described in the following sections.

2.2.2.1 Principles of Existing Hardware Monitoring Systems

2.2.2.1.1 Data Collection Methods

All hardware monitors rely at least partially on being able to tap into various busses, flag lines, interrupt vector locations, and controllers by means of high-impedance probes which "mimic" the logic state of these points, and report them to the hardware monitor logic. As such, they are

entirely passive components from the standpoint of the system being monitored. Between 18 and 144 such probe points can be under surveillence by the monitoring equipment, although only half of these may be "active" at a given instant. The probes are capable of responding to pulses as short as 10 nanoseconds, and can sustain repetition rates of 40MHZ (one pulse every 25 nanoseconds).

Probes are attached to probe points with "temporary" connectors (i.e., spring clips). The points chosen for interrogation are generally single control status lines (i.e., CPU idle, channel request). Meaningful events in terms of system behavior are rarely measureable by a single sensor. These events are generally defined by ordered combinations and sequences of sensor-detected signals. Both the COMTEN and TESDATA systems rely on a "plugboard" arrangement where sensor inputs may be combined by means of standard gate logic (AND's, OR's, NOT's, etc.), to define system "events", which are the basic units of information for processing by the monitor's hardware/software systems.

A further use of the plugboard "patch panels" is to direct the measured "event" to collection and/or distribution circuitry, where the event will be either "counted", "measured", or "mapped". These terms will be discussed in the following section.

2.2.2.1.2 Collection of Event Data

Collecting information about the frequency and duration of events, and about the contents of information transfers is accomplished by comparator and collector circuitry within the monitor. A detected event can be "patched" to a counter circuit, which will record the number of times the event occurs. In addition, clock circuitry can be connected which accumulated the amount of time for which the event exists. Some models permit "time stamping" the event with the actual time of day.

In addition to events, actual data values may be recorded (for instance, addresses as they appear on a bus). Generally, the sampling of this sort of information is dictated by the occurrence of a logical event. Once sampled, the data can be held (buffered) and submitted to comparater logic, which will determine:

- where to store the data, or
- if it is within a desired range, or
- equal to a specific value.

This determination may lead either to the data's storage or its rejection.

2.2.2.1.3 Data Storage and Real-Time Reporting

The method of storing the data collected by monitor procedures described thus far is the prime point of departure between COMTEN's and TESDATA's monitoring. The COMTEN systems record the data with little further processing. Aside from perhaps allowing display of the contents of specific frequency counters by L.E.D.s, the COMTEN system provides very little immediate information to personnel supervising the monitoring procedures. Instead, the collected information is spooled onto magnetic tape at set intervals for later, off-line processing. The COMTEN system can be thought of as a passive system peripheral.

The TESDATA system are all built around powerful minicomputers, and are considered a node in a distributed processing system. The minicomputer is equipped with a tape or disk-based operating system allowing user interaction with the monitoring procedure. Without repositioning

sensors, the user may load new monitor procedures, query the monitor "data base" for dynamic reports, or may specify the dynamic display of monitoring results on either a special bar graph display unit or a more conventional CRT. In addition, dynamic data exception reporting is possible because of the system's ability to sense, analyze and record data simultaneously.

The most advanced of the TESDATA systems can serve as the central node in a distributed (multiprocessor) monitoring network. Secondary measurement systems transmit data to the primary system over voice-grade lines, permitting effective analysis of remotely-distributed systems.

2.2.2.1.4 Data Reduction and Report Generation

2.2.2.1.4.1 COMTEN

Both representative manufacturers offer extensive software report generation packages. COMTEN offers them unbundled, as it tends to have many packages tailored for specific IBM operating systems. COMTEN software must run on the host computer, relying on the tape history generated by the monitoring system for data input and the host's hard copy facilities for output. These programs do not appear to be at all interactive. The user is able to define the time interval of data reporting. Beyond that, report formats appear quite static. Standard reports generated by basic COMTEN software (DYNAPAR) include:

- o Counter Processor

  - Counter summaries
  - Profiles - percentage of times that various monitored events occur during a period (horizontal bar graph)
  - Counter statistical analysis (mean, standard deviation, min, max)
  - Kiviatcharts
  - Histograms

- o Distribution Processor

  These show the relative frequency of activity in various parts of a hardware resource.

- o Disk Analysis Profile

  An information package for evaluation of data set placement.

- o Data Path Analysis

  This package provides information for configuration management by reporting on utilization of channel and bus activity, generally for a multiprocessor environment.

- o Operating System related reporting

  MVS Performance Measurement/Memory Mapping report package.

COMTEN reports, contrary to their publicity literature, appear to be of direct use only to technical management, and would require annotation/interpretation to be useful to management not directly involved with the computer facility.

2.2.2.1.4.2 TESDATA

TESDATA's facilities for reporting monitor findings to the user are considerably more comprehensive than can be offered by COMTEN because the monitoring instrument is controlled by a minicomputer with a multi-user interactive operating system. All software supplied for the presentation of reports runs on the monitor's minicomputer, under control of the operating system, simultaneous with the collection of data. This dynamic monitoring capability supports a number of TESDATA peripherals that can continuously report to machine-room management on the current operating efficiency of the system. One such peripheral is a standard alphanumeric CRT, while the others, a bar-graph display and a large-character 32 character alphanumeric plasma display unit, can report utilization and exception conditions from some distance.

In addition to supplying its dynamic output capability, the TESDATA operating system allows the user to dynamically invoke all report generation programs. The user can supply a variety of input parameters to these programs, and direct outputs to either hard or soft copy devices, eliminating paper waste. Because the report formats can be interactively tailored to fit the user's requirements, the reports are far easier to interpret than the COMTEN reports, and can more readily be understood by management of all levels.

TESDATA systems do "spool" incoming information to mass storage devices (tape or disk), so that reports can be requested for any given time period contained in the system "data base".

TESDATA can supply turnkey report generation packages (T-PACs) for their entire line of machines, although the larger models in the series are capable of supporting development of a custom monitoring package. Included with the T-PACs are plugboard and sensor hookup configurations. The available T-PACs are:

- o Basic System Profile
- o Regional Mapping
- o Advanced System Management Profile, including high speed buffer activity and effective channel transfer rates
- o Disk Contention Profile
- o Regional Disk Mapping and Disk Arm Motion Profile

In addition to the T-PACs, TESDATA can supply an optional MVS-SMF support package, which allows the user to combine hardware monitoring information with output from IBM's SMF (Software Monitor Function) to produce a report linking control information previously only available in separate reports.

2.2.2.2 Typical Hardware Monitor Configurations

Figures 2-3 and 2-4 are block diagrams of the two typical hardware-based commercial monitoring systems discussed. While it is generally accepted that the TESDATA monitoring systems are the "Cadillacs" of hardware monitors, COMTEN does offer more sophisticated systems than the basic monitor presented here.

The TESDATA minicomputer based monitoring systems are considerably easier to configure, and requests for new kinds of reports can be serviced more easily than in the COMTEN systems. The manufacturers' interfaces with the host machine are essentially the same; where they differ is in the point where the

monitored data must be stored. COMTEN is content to spool their data sequentially for offline processing, whereas TESDATA in effect adds an entire system to the monitor, including a data base management facility.

If the kind of information that needs to be monitored does not change often, and there is no need for dynamic condition or exception reporting, then the COMTEN approach to "batch" monitoring is cost effective. For complex systems, where on-line reporting capabilities are essential, the much greater cost of the TESDATA systems is justified.

The hardware monitors discussed so far have been used in the optimization of large mainframe systems. They tend to be very expensive utilizing fairly large minicomputers as integral components.

At this point in time, except for the Mc^2 hybrid monitor, no scaled-down system with economics sufficiently attractive is available for the AN/GYQ-21(V).

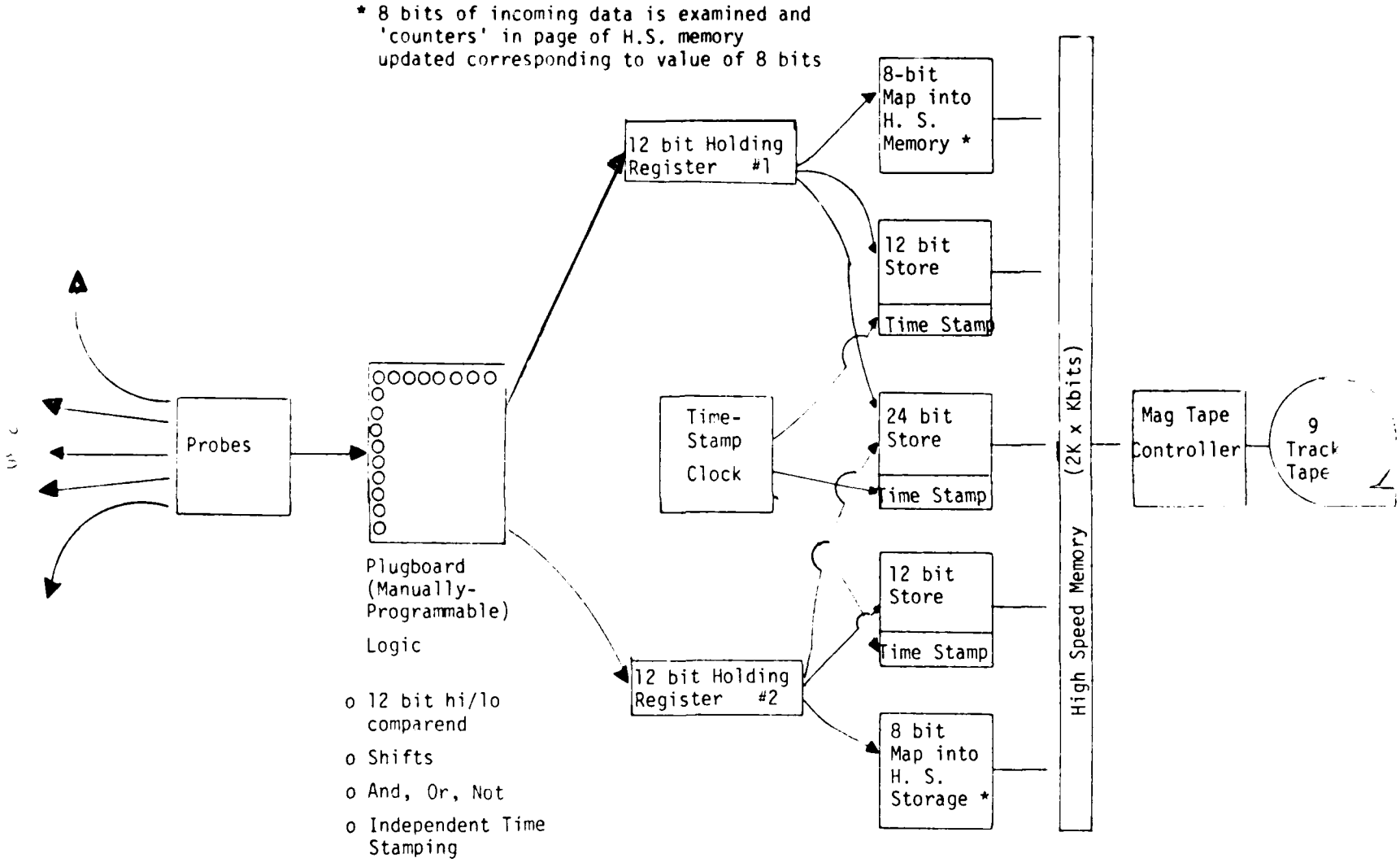
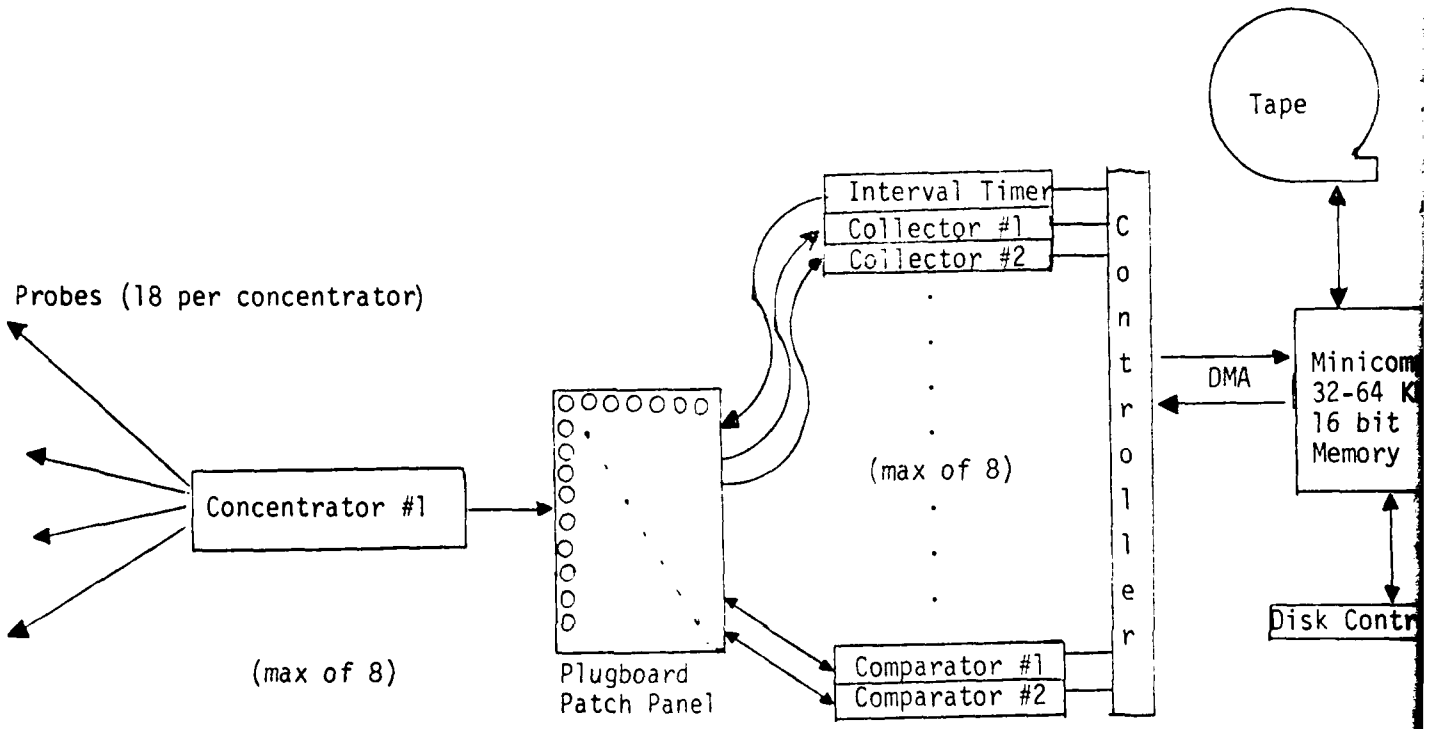


Figure 2-3 COMTEN Basic Monitor Design

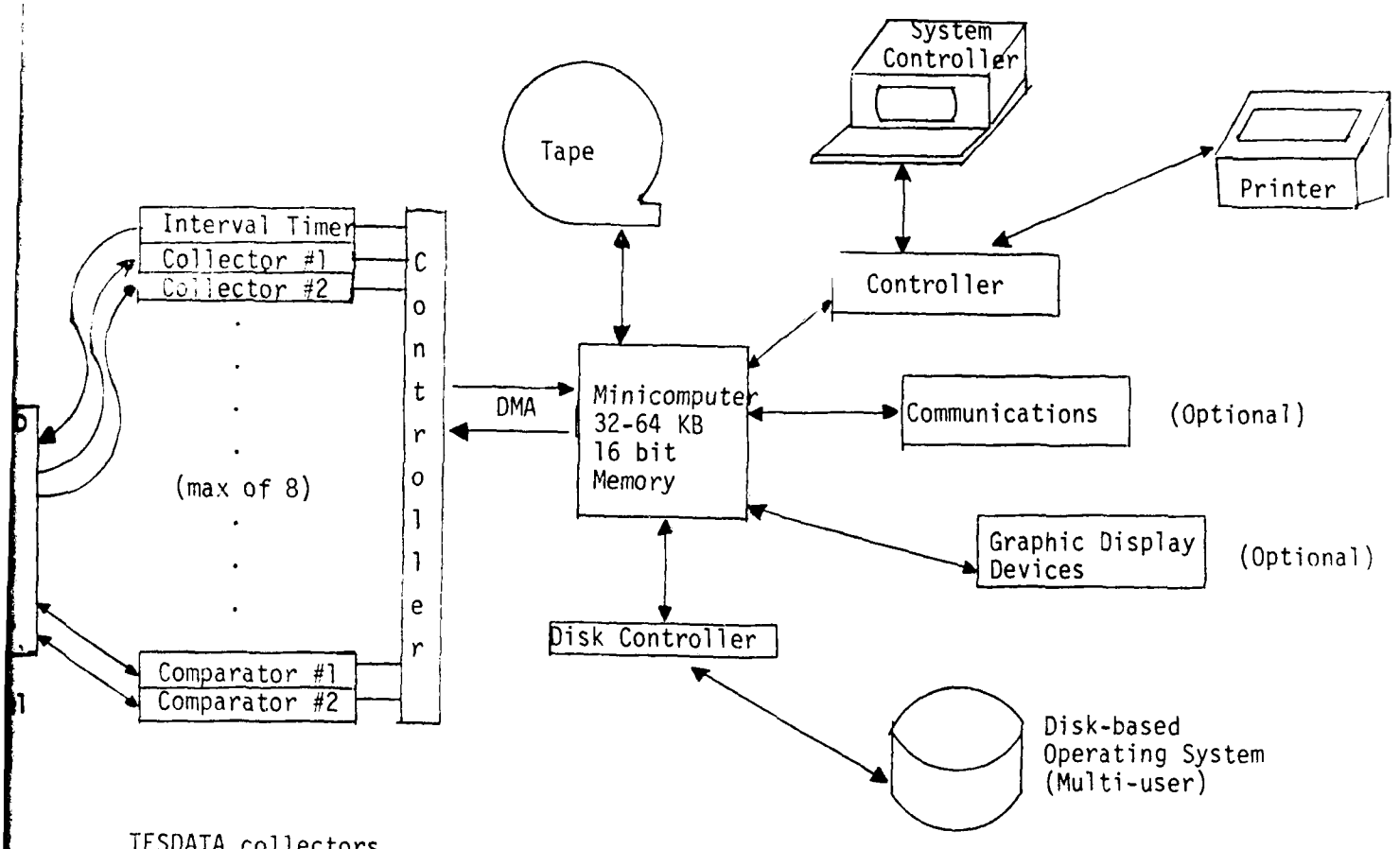


TESDATA collectors
do the following:

- o count
- o store
- o time
- o map
- o time-map
- o time-store

The functions of each collector are software-selectable from the system console

Figu



TESDATA collectors do the following:

- o count
- o store
- o time
- o map
- o time-map
- o time-store

The functions of each collector are software-selectable from the system console

- o TPAC data collection/reduction report generation software
- o User-defined data collection/ reduction, report generation software
- o Monitoring System Self - Diagnostics
- o Communications software

Figure 2-4 TESDATA High-End Monitor Design

improper execution sequence, addressing out of bounds, and improper data capture from an external device. They are useful for detecting but not diagnosing transient hardware malfunctions. To locate these problems, the faster clocking speed of the logic timing analyzer is necessary. Typically, a timing analyzer must run at 4 to 10 times the speed of the hardware being monitored to examine the behavior of the hardware between its own machine cycles.

2.2.3.1 Basic Functions

Logic analyzers generally have adjustable voltage threshold limits between a logical "0" and a logical "1". As the analyzer deals with many different logic "families" (i.e., TTL, TRL, DTL), it allows oversetting the threshold to trap marginal logic conditions, when a signal swings close enough to the threshold.

Available analyzers range in sample frequency from a few megahertz to 200 Mhz. The faster the analyzer, the higher the price, on an approximately logarithmic scale. For example, a 5 to 10 Mhz analyser sells for about $1500, a 50 Mhz analyzer sell for about $4600, and a 200 Mhz analyzer costs about $20,000. Part of the greater cost of the faster asynchronous units can be explained by their larger memories.

One feature often present on asynchronous timing analyzers that increases their utility is "latch mode". Latch mode allows an analyzer to capture data spikes or events that are narrower than the sample clock interval: the "latch" captures the transition and "holds" it until the next clock cycle.

In this way. the analyzer captures the high-speed anomaly and stores it in its proper time sequence in relation to the rest of the data. This, in effect, extends the bandwidth capability of a lower speed analyzer.

All general-purpose logic analyzers have the ability to "trigger" on the basis of the combinational states of several logic lines. A trigger variable canbe constructed from a sequence of logical '1', logical '0', or 'don't care' conditions. Lines being monitored can be compared to this trigger variable, with a 'match' condition beginning the recording process described earlier. Added to this capability is the ability to trigger when a 'match' condition fails to exist as well.

2.2.3.2 Special Monitoring Devices for the PDP-11 Series of Computers

- o Formation

  Formation F801 Program Debug Console for PDP-11 users can be set up to stop or interrupt on UNIBUS receipt of a predetermined memory or interrupt location. Upon halt, the F801 contains the last 16 addresses, in order, to appear on the UNIBUS.

  In addition, the F801 can respond to a match of pre-set address to UNIBUS address by generating a UNIBUS interrupt, which is handled by user-written software. The interrupt vector address and bus request level are set by means of jumpers on the control electronics plug-in module.

  Both of these features can be executed:

  (a) instructions accessed in "write" mode.

  (b) instructions accessed in "read or write" mode.

The 801 mounts in a 19" rack or DEC cabinet, is 5 1/4" high, draws 1.5 amps @ +5 volts, and uses one standard UNIBUS load. The console sells for $1,200.

o Three Rivers Computer Corporation Bus Monitor.

Three Rivers Computer Corporation's Bus Monitor is designed to assist hardware and software personnel in production and maintenance of their PDP-11 based systems. All 56 UNIBUS signals are displayed with LED indicators with switch selectable FOLLOW or HOLD mode on the address and data lines. HOLD mode is controlled by the 9 position function select rotary switch. Eighteen three-position address switches allow the monitor to display only the activity that occurs on a single address or group of addresses. UNIBUS operation may be suspended by such events as a memory reference, a write to a device register, an interrupt, a parity error or an externally supplied signal.

The basic Bus Monitor may be enhanced with the Examine/Deposit option. This adds the ability to examine or deposit into any memory location or bus accessible register. The Repeat/Repeat-Next functions for examine and deposit allow one to create high levels of NPR traffic, to search for parity errors, or to fill memory with a constant. The repeat rate is variable from once every 5 seconds to 400,000 per second. With the Examine/Deposit option the Bus Monitor may also be set up to interrupt the CPU on the event selected by the function select rotary switch of the basic monitor. Interrupt vector and bus priority level are set by the user. Function select outputs and inputs are provided the interconnection with logic analyzers, digital counters and oscilloscopes. The monitor sells for $995, or $1500 with the Examine/Deposit option added.

- Hewlett Packard PDP-11 UNIBUS Interface

  Designed as an accessory for Hewlett Packard logic analyzers, the device provides a method for fast, easy connection of a logic analyzer with a PDP-11 minicomputer. It consists of of a quad-height board, which plugs directly into a PDP-11 SPC (Small Peripheral Control) slots, and allows access to all 56 signals on the Bus. Circuits on the board generate a clock signal for the logic state analyzer. In addition, switches on the board provide qualification of UNIBUS activity, so that reads, writes, interrupt vectors and DMA transfers can be selectively captured for detailed analysis. The price is quite modest at $300.

2.2.4 Hybrid Monitoring Concept

A hardware monitor is a device external to a measured device, that senses electronic signals in the circuitry of the measured system and processes them externally to the system in question. Several concepts are critical to understanding the distinction between hardware and other types of monitors.

The terms "passive", "active", and "cooperative" as used below refer to the three primary data collection modes available for use in a 21(V) monitor:

- Passive measurements are those requiring no monitor processing logic or sequencing other than that necessary to detect a signal from the probes and interfaces and to convert these signals into event counts, distributions and inter-event intervals.

  A passive monitor can collect data at the tested system's cycle speed or at a rate determined by some external event (e.g., monitor cycle time, monitor clock, or frequency divided host clock).

- Active measurements are those that result from specific monitor activities that "request" information from the host or otherwise alter (temporarily) the host's status. An example of this technique is the transfer of the program counter (PC) from the host CPU to the monitor via the UNIBUS.

- o Cooperative measurements are those requiring a "handshake" between host activities and monitor resources. A simple example is the case where programs being debugged can utilize the monitor (through one or more UNIBUS addresses) to record specific events and counts.

A distinction can be made between the following major classes of hardware monitors - the wired program monitor, the stored program monitor, and the hybrid monitor.

2.2.4.1 Wired Program Monitors

Wired program monitors are characterized by their completely passive mode of measurement; i.e., such a monitor merely records events that it has the capacity to directly measure. Because of its limited "intelligence", this type of monitor is capable of reporting only event trace and event summary data for a limited number of events that must be defined prior to test (no dynamic changes to event definitions can be made).

In its simplest hardwired configuration, the wired program monitor could be similar to that illustrated in Figure 2-5 (Ferrari, 1978). The illustrated tool only measures the utilization and overlap factors of CPU and I/O channel by observing the busy/idle and problem/supervisor flip-flops of the CPU and the busy/idle flip-flop of the I/O channel (presuming they are accessible for measurement); it only provides summary data since no provision for trace recording/playback has been made.

In a more complex configuration, the wired program monitor can be equipped with a "patchboard" that would allow the various probes to be logically combined in other ways so that different measurements could be made. Patchboards

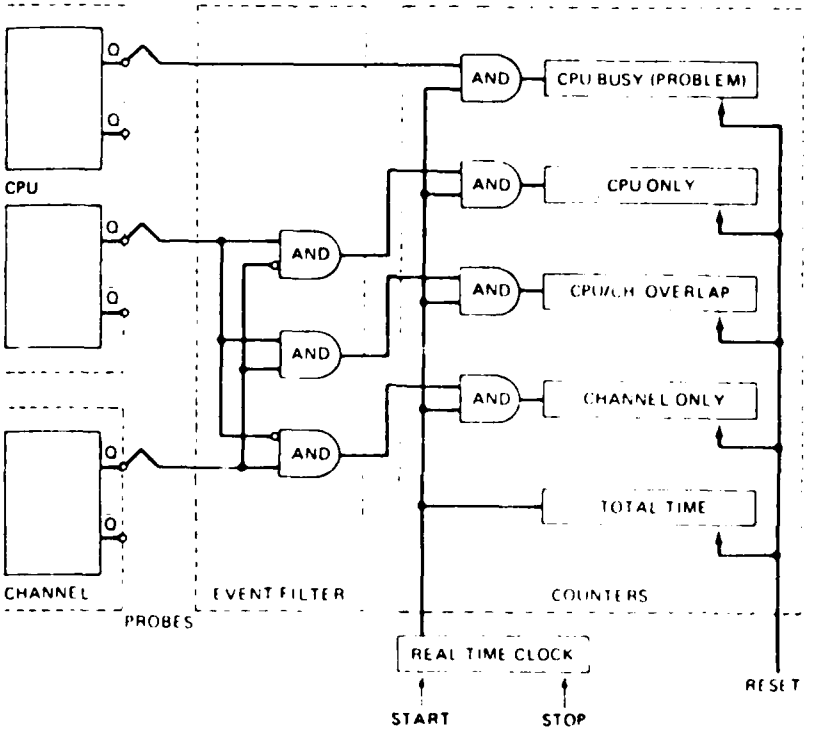


Figure 2-5 Wired-Program Monitor

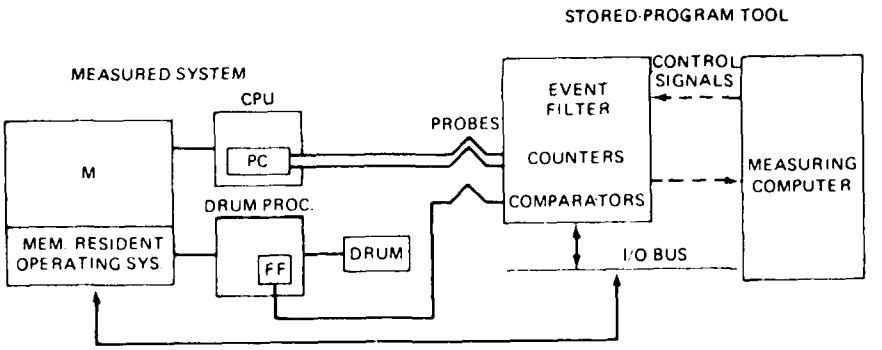
supplied with commercial units are sophisticated enough to allow detection and counting of sequences of events.

2.2.4.2 Stored Program Monitors

The introduction of mini and micro computers have made it possible to design monitoring configurations in which the user can determine to a larger extent what kind of measurements should be taken. Figure 2-6 (Ferrari, 1978), shows a configuration in which the program counter of the CPU and the busy/idle flip-flop of the drum are being observed by a monitoring device. In addition, information is obtained from the operating system tables to further sub-classify the observed events as they trigger the monitor by a signal from the event filter. This is achieved through the additional connection between the measuring computer and the measured system's I/O bus.

Although such connections significantly enhance the discriminatory power of the stored program monitor, they introduce some measure of artifact; the monitor is actively pursuing the collection of data from the measured system through utilization of the measured system's resources (its I/O bus). The stored program monitor's processor element is capable of preprocessing and storing more complex measurement data so that a better picture of the system's performance can be screened.

Lastly, the availability of buffering and "intelligent" I/O control within the stored program monitor offer enhanced capabilities for the capture and recording of more detailed test information over longer periods of time.



PC - Program Counter of the measured system

FF - Busy/Idle Flip-Flop of the drum processor

Figure 2-6 Stored Program Configuration

2.2.4.3 Hybrid Monitors

A hybrid monitor is a cooperative and mutually synchronized configuration of an internal software monitor and a hardware monitor. It combines the best of two worlds:

- A hardware monitor rapidly senses a variety of events, but is limited in the ability to detect the stimulus for a set of events.
- A software monitor is able to relate events to the stimulus of the events, but is limited in the ability to spend time processing that information.

In a hybrid configuration the hardware monitor is basically treated as an intelligent peripheral device.

It is sometimes hard to make a clear distinction between the equipment of the stored program hardware monitor and a hybrid monitor. Figure 2-7 shows a hybrid system where the connection between the data channel and the minicomputer provides for two-way communication between the hardware monitor and the software monitor in the host system; either can interrupt the other. The result is that causal relationships between events can be established. In addition, the minicomputer provides for opportunities to:

- Assist in setup and reconfiguration of combinatorial logic of the event filter.
- Off-load the software monitor components on the host system.
- Produce on-line reports and detailed summary reports.

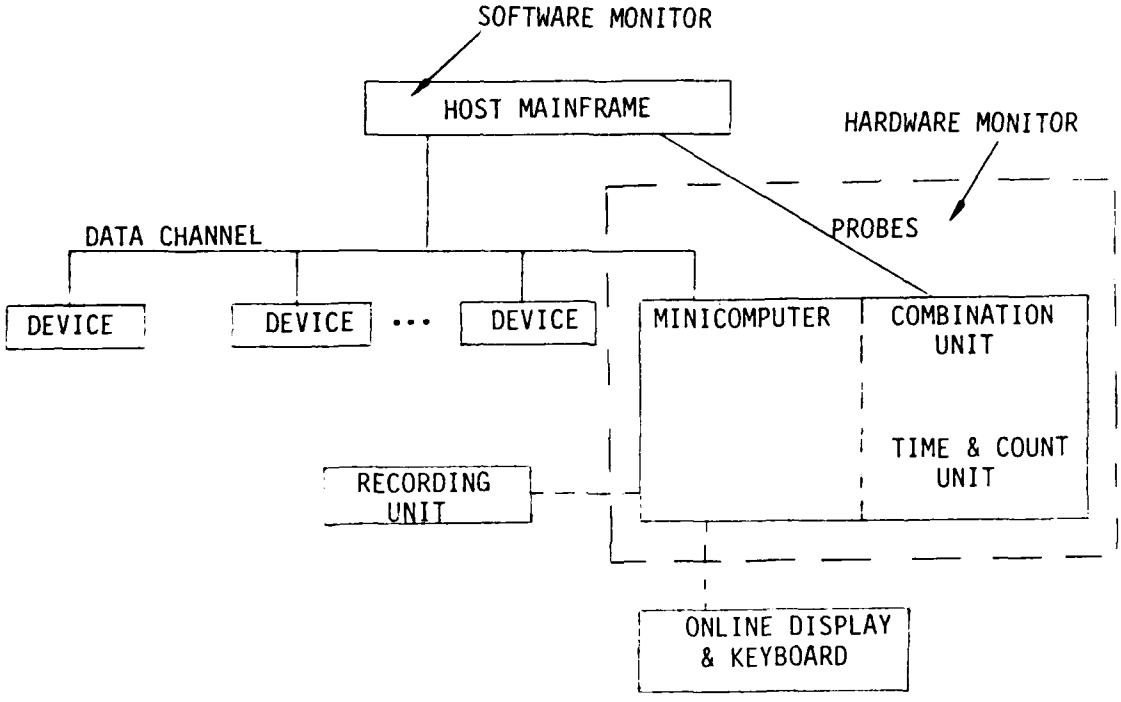


Figure 2-7 A Hybrid Monitor

3.0 HARDWARE DESCRIPTION

This section presents the hardware concept for the microprocessor based AN/GYQ-21(V) monitor. The basic concept consists of a set of modules whose functions parallel the classic data collection through recording sequence.

- o Raw data sensing/collection
- o Event detection (preprocessong of raw data)
- o Event processing and recording

Four modular hardware systems are defined for the hardware design concept.

- o Sensing and high speed preprocessing subsystem (Vector Generator)
- o Medium speed event subsystem (Programmable Patchboard)
- o Extended event processing subsystem
- o UNIBUS Interface Subsystem

All subsystems operate under the general control of a "basic" monitor microprocessor. Each subsystem may vary considerably in its complexity from a "basic" configuration that still provides significant amounts of performance data at minimum cost and installation complexity to a "fully loaded" configuration that will simultaneously service the many diverse requirements of programmers, system designers, system integrators, installers, maintenance personnel, and managers.

The design concept recognizes the many orders of magnitude difference between the times in which CPU and peripheral control signals change (nanoseconds) and the times in which statistically relevent system-wide evaluation aids such as average responsiveness are calculated (minutes or hours). To accomodate this difference, four levels of detection/processing are necessary.

- Level 1: Changes that occur at rates faster than the basic machine memory cycle time. In this domain, for example, are 21(V) UNIBUS handshaking signals that determine the latency of bus acquisition, slave response, and change of bus master status between CPU and peripheral devices. Typical times are from 10 nanoseconds to less than 500 nanoseconds.

- Level 2: Changes and status indicators that reflect time durations on the order of one machine memory cycle. Among the most important data elements at this level are the bus address and data lines and UNIBUS transaction class (interrupt, NPR, etc.). These events typically occur at .05 to 1 megahertz.

- Level 3: Events and status indicators that change at rates significantly slower than a single memory cycle but which could, for example, reflect primary changes in CPU operational status and peripheral device activity. Clear examples of events at this level are:

  - A change in processor priority
  - A change in processor mode

- Completion of a block transfer

  Characteristics times at this level are in excess of 10 microseconds and more typically in excess of 100 microseconds.

o Level 4: Human perceivable events measured in seconds to hours.

Each of these levels contributes valuable information to the higher level event detection and processing.

The major components of the complete monitor and its AN/GYQ-21(V) interfaces are shown in Figure 3-1. Events of all frequencies occur in the 21(V) host to the left of the figure. The high speed events are preprocessed and converted to medium speed signals by the vector generator that is physically connected to the host. Medium speed signals enter the programmable patchboard (PP) where, per instructions left by the monitor microprocessor, they are AND'ed, OR'ed and otherwise subjected to data reduction that results in incrementing of counters or in self-modification of the PP-program to allow sequence detection and interval stop/start. The PP interrupts the microprocessor when program selected level 3 events have occurred. These events are handled by microprocessor code which eventually results in the storage (for later retrieval) of only desired measurements.

Data flowing from left to right in Figure 3-1 arrives at the PP from the vector generator subsystem. The data from the latter subsystem is all obtained in a "passive" mode, that is, the monitor just processes what it detects with no attempt to influence the host. Data from the special UNIBUS interface subsystem, however, is based upon active and cooperative measurement techniques that force the host to give up the desired information. At desired sample times and under ultimate microprocessor

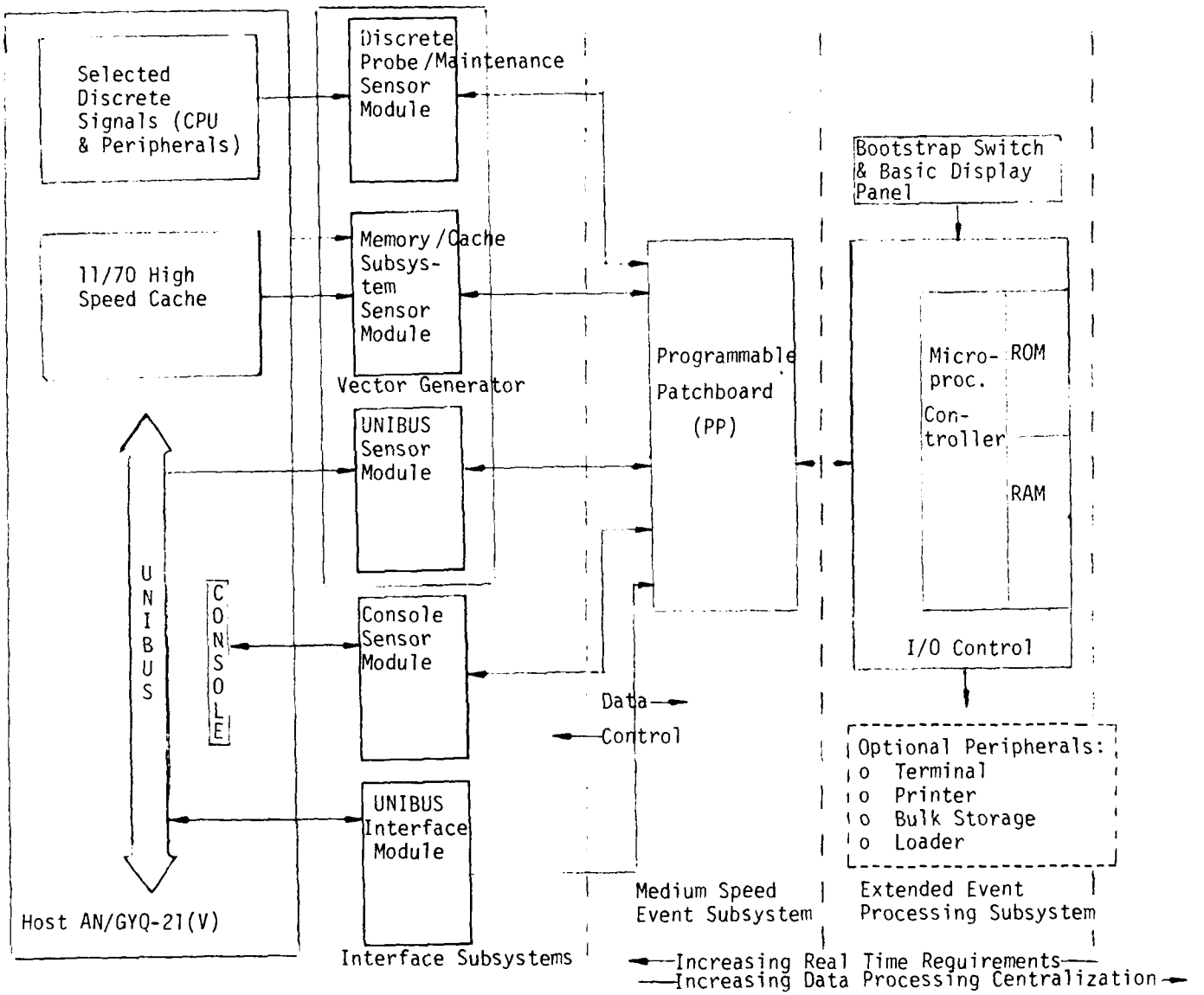


Figure 3-1 Monitor System Design

control, active measurements are initiated by creating appropriate signals on the host (e.g., interrupt request, non-processor request, etc.) to cause either the transfer of host data elements to the monitor or the changing of the host execution mode (stopping the CPU, single stepping it, etc.).

When monitoring a multiple AN/GYQ-21(V) system several vector generator and programmable patchboard pairs might be involved delivering their respective measurements to a common microprocessor (or several microprocessors, as the need may be). The central microprocessor(s) system will perform the correlation of events observed in the total system under observation.

3.1 AN/GYQ-21(V) System Description

The AN/GYQ-21(V) is a software and (to a large degree) hardware compatible series of computers and computer peripherals that cover the Digital Equipment Corporation PDP-11 series plus peripherals and software from other suppliers (such as Bunker Ramo). For this reason, maximum utilization will be made of such commonality factors as:

- The 21(V) operating systems (RSX-11 and IAS) are multiprogrammed and multitasked. They are heavily interrupt ("real time") oriented. Maximal advantage can, therefore, be taken of system hardware and software interrupts (traps) to signify changes of system state.

- The 21(V) operating systems are localized in low memory and with the possible exception of common subroutines, they execute in "kernel" mode. This information is highly useful in developing evaluation factors such as CPU time in executive service (overhead) and utilization of common routines (time spent in linking and unlinking dequeues, etc.).

- The 21(V) design provides for easy accessability of the CPU's general registers, memory mapping parameters, processor status, and cache statistics, among other variables, via a straightforward existing UNIBUS interface.

As the (V) in the nomenclature indicates, no two systems are the same. They vary in memory size, CPU type, CPU options such as hardware floating point, memory management, clock type, cache type and size (if any), I/O bus structure and manner of memory access, peripheral mix and relative positions, I/O bus acquisition priority, software operating system, and application program mix and processing load. While the mainframe in a

21(V) can be any of a series of eight different machines, they are similar in the following ways:

- All have the same instruction set
- All use the UNIBUS for I/O device control and status reporting
- All but the PDP-11/70 use the UNIBUS exclusively for data exchange
- Except for the PDP-11/70 mass bus controller, all have similar peripheral devices
- All have similar (but not identical) card module and cabling structures and air flow and physical access arrangements
- All current 21(V)s have similar control consoles except for differences which are irrelevant as far as the hardware is concerned

As a result of the above, the discussion of the 21(V) emphasizes the following four aspects for the reasons stated.

- UNIBUS - It is a window to the heart of 21(V) operation and performance.
- 11/70 High Speed Memory and MBC Buses - This bus complex, in addition to the UNIBUS, allows most aspects of every 21(V) transaction to be monitored and measured.
- CPU/Console Interface - A set of important CPU status and control data is readily accessible at this interface.

- o Restrictions and Limitations - A common set of architectural, physical and operational restrictions apply to all 21(V) systems.

3.1.1 UNIBUS

In AN/GYQ-21(V) systems most of the computer system components and peripherals connect to and communicate with each other over a common bus known as the UNIBUS. Address, data, and control information are transmitted on the 56 lines of this bus. The discipline required to communicate is identical for every device connected to the UNIBUS. Each device, including memory locations, processor registers, and peripheral device registers is assigned a unique address. Communications on the UNIBUS are bi-directional and asynchronous making it compatible with devices operating over a wide range of speeds.

Any device can be easily connected to the UNIBUS by utilizing a connector which accomodates the standard UNIBUS cables which are employed to interconnect the components of the system.

The signals which comprise the UNIBUS afford an extremely comprehensive set of parameters for evaluating system performance. In particular, the duration of and interval between various control signals are of prime importance in determining the degree of optimization present in the system configuration both from a hardware and software standpoint.

Communication between devices on the bus takes the form of a master-slave relationship. The device which currently has control of the bus is referred to as "bus master". The device with which the master wishes to communicate is defined as the "slave". These relationships are dynamic.

The processor may, for instance, pass bus control to a disk. The disk, as bus master, could then transfer data to/from a slave memory bank or other peripheral without CPU intervention.

Communications on the bus are interlocked such that each control signal issued by a bus master requires a response from the slave in order to complete the transaction. Communication is thus independent of the physical length of the bus and response times of the various devices. No time-dependent interfaces exist for the UNIBUS, thus no pulse-width or rise-time restrictions are imposed on devices attached to the bus.

Any device which wishes to use the UNIBUS to make a DMA type of data transfer activates the "Non-Processor Request"(NPR) signal line. Whenever the arbitration logic determines that the processor does not require the bus it issues a "Non-Processor Grant" (NPG) in response to the pending NPR. Since NPG is a cascaded (daisy chained) signal, the first device on the bus which has its NPR active will intercept the NPG and issue "Select Acknowledge" (SACK) informing the arbitration logic that the grant has been recognized. The interval between NPR and SACK represents the bus acquisition time. By measuring the interval between NPR and SACK, it is possible to determine the effectiveness of the arbitration logic in responding to the various devices attempting to utilize the UNIBUS.

To allow for data or control (e.g., interrupts) exchanges with the CPU, a set of Bus Request (BR) and Bus Grant (BG) lines are provided for use by I/O devices to acquire bus mastership. These functions are electrically and functionally the same as the NPR/NPG lines described above except for the priority level at which they operate. NPRs can stall the CPU in mid-instruction execution and are the highest priority in the system. BRs are only effective between CPU instructions. Figure 3-2 shows the priority levels relative to internal software and CPU priorities.

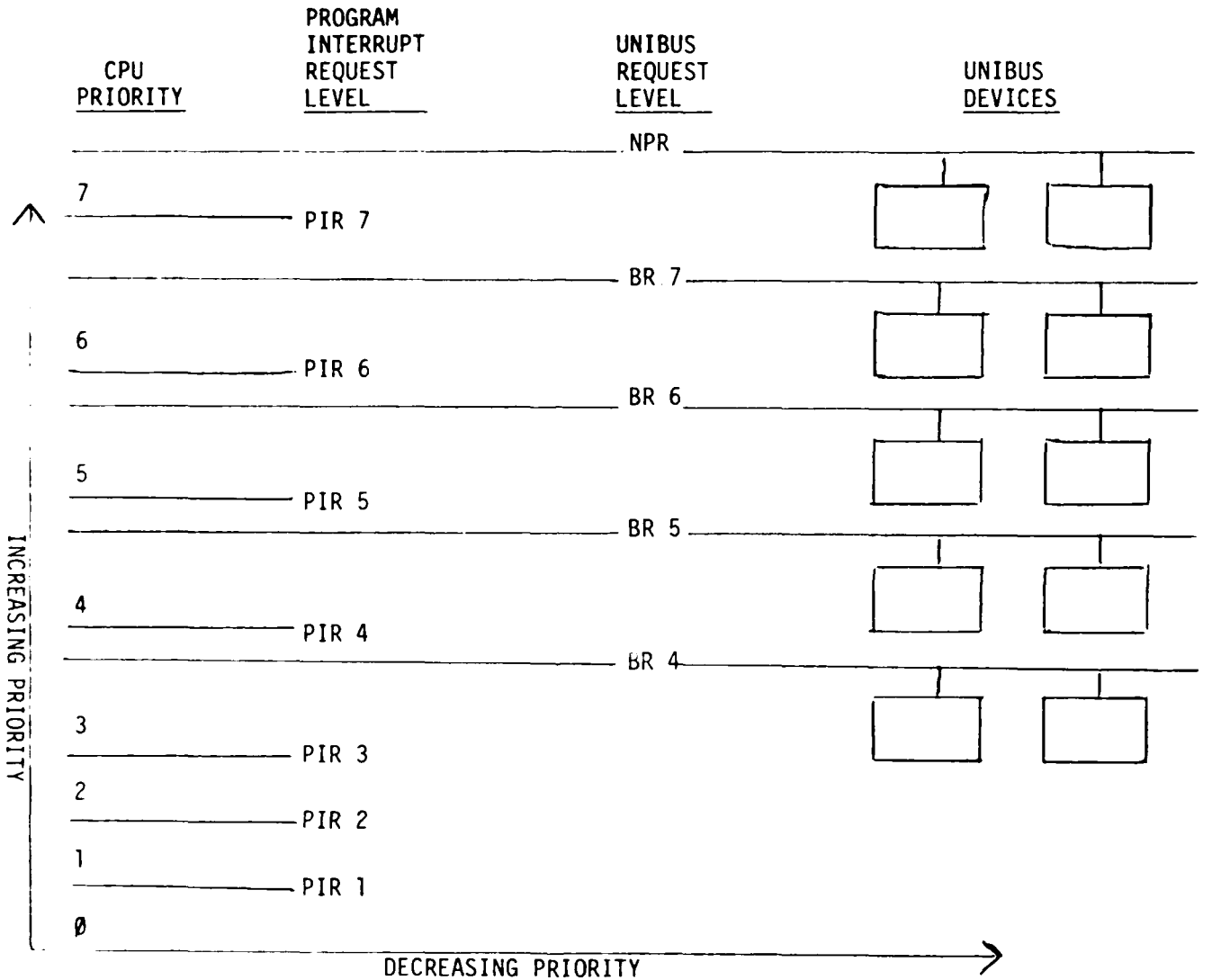


Figure 3-2 UNIBUS Priority Arbitration

After acquiring the bus a device waits until the previous bus master relinquishes control of the bus by clearing "Bus Busy" (BBSY). The new master then activates this signal and proceeds to execute the desired data transfer. The duration of BBSY is a direct measure of the length of time that a device occupies the UNIBUS.

Once a device has acquired control of the UNIBUS, it issues a signal called "Master Sync" (MSYN) along with the appropriate address of the device or memory location it wishes to exchange data with. In response to MSYN the requested device generates "Slave Sync" (SSYN). This constitutes the required handshake relationship needed to consummate data transactions over the UNIBUS. The time interval between MSYN and SSYN is a significant measure of the performance of the UNIBUS in general and selected devices/ memory modules in particular. Since systems frequently have many memory modules (as well as bus repeaters, etc.) the relationship between the physical location of devices on the distributed UNIBUS can significantly effect system performance. The address associated with any particular bus transaction which is inordinately long identifies the device or block of memory which is responsible for compromising system performance.

3.1.2 PDP-11/70 High Speed Bus

Many current versions of the AN/GYQ-21(V) use the PDP-11/70 as the main frame. These 21(V)s also have a UNIBUS but the high speed peripherals use a second bus for their data exchanges with memory. The UNIBUS is still used for overall I/O transaction control and status reporting, however, and the preceding UNIBUS discussion applies equally to these 21(V)'s as well.

The PDP-11/70 has an expanded interval implementation which improves system thruput by virtue of the high data rate memory bus with its provisions for up to four high-speed (MBC) I/O controllers. Unlike the other PDP-11 systems, main memory does not reside on the UNIBUS and hence a PDP-11/70 UNIBUS is rarely heavily loaded. This additional capability is intended for use with large capacity disk and magnetic tape systems which would otherwise occupy the UNIBUS to an excessive degree.

The 11/70 has a cache memory which enhances program execution since information stored in the cache can be accessed much more rapidly then from main memory. Because of its function and position relative to the other functional components of the system, the cache acts as a clearing house for all main memory accesses. Three sources of requests for main memory are possible: processor, UNIBUS Map, and the MBC I/O controllers. System performance on the 11/70 is most easily monitored by examining the requests and responses of the cache control logic which allocates memory cycles on a priority basis to the three contenders. The performance of the cache in executing its primary function can be evaluated by examining the input to HIT/MISS Register which indicates whether the six most recent program memory references were contained in cache. If a low percentage of hits is occurring, the programs being executed are not optimally organized and corrective action is indicated. Both monitoring of hits and disabling of the cache can be accomplished by accessing special cache registers via standard UNIBUS addresses (17777746 and 17777752).

MBC requests for service are handled by arbitration logic located on the cache modules. Since there are four such interfaces, the arbitrator must decide the order in which to service these requests. Jumpers installed in each of the four RH70 interface locations determine the

priorities assigned to each device. Monitoring the request lines from each channel and the arbitrators response of "select address" back to the controller gives a figure of merit as to the thruput from each high-speed device connected to the 11/70.

Although high speed devices exchange data on the high speed bus, they still communicate control information over the UNIBUS and generate processor interrupts via this bus as well. Consequently, system performance can still be evaluated by monitoring the intervals between transfer of control information to a device and the consequent interrupt from that device after the requested transaction is complete.

The final unique aspect of the PDP-11/70 is its Unibus Map Facility (UMF). The function of the UMF is to allow UNIBUS resident devices to perform DMA accesses to any of the 11/70's memory locations. A total of 31 mapping registers (UMR) are provided. Each UMR allows access to a 8K byte contigious segment of memory. The UMRs are shared by all of the UNIBUS devices. They are allocated and loaded by software on a dynamic, per transaction basis.

3.1.3 CPU/Console Interface

The CPU/Console Interface is significant to the study because it provides convenient access to a number of important monitor data elements as well as providing a mechanism by which, in the active mode, the hardware monitor can exert real time physical control over the 21(V) system if desired. The CPU consoles are interfaces with the CPU and the remainder of the system via a pair of ribbon cables. The data/control elements of interest provided at this interface are the following:

- o Address Display Indicator Inputs
  - User I and D
  - Super I and D
  - Kernel I and D
  - Program Physical
  - Console Physical
- o Mapping Mode Indicator Inputs (PDP 11/70 only)
  - 16 bit
  - 18 bit
  - 22 bit
- o Data Display Indicator Inputs
  - Current Data Path
  - Bus Register Contents
  - Current CPU ROM and Floating Point Processor Address
  - Display Register Contents
- o Processor State Indicator Inputs
  - RUN: CPU is executing instructions
  - PAUSE: CPU is waiting for UNIBUS device or memory
  - MASTER: CPU is current UNIBUS master
  - USER: } { CPU is executing
  - SUPERVISOR: } { program instructions in
  - KERNEL: } { the indicated mode.
  - DATA: The last memory reference mode was to Data (D) space rather than Instruction (I) space.

- Error
  - Address Error
  - Parity Error (PDP 11/70 only)
- Address Select Switch Group
  - Virtual: 6 position
  - Console Physical
  - Program Physical
- Data Select Switch Group
  - Data Path
  - Bus Register
  - μ address of FFP/CPU ROM
  - Display Register
- System Control Switches
  - Load Address
  - Examine
  - Deposit
  - Continue
  - Enable/Halt
  - Single Instruction/Single Bus Cycle
  - Start

3.1.4 Multiprocessing AN/GYQ-21(V)s

Within a site, individual AN/GYQ-21(V) units can be linked together to create a single operational entity, with each unit contributing its designated tasks to the overall system operation. This technique of "distributed processing" permits each AN/GYQ-21(V) unit to compute simultaneously with other units, accessing its own or other data bases on a priority basis, resulting in a net performance more powerful than most large-scale machines. As workload requirements change, AN/GYQ-21(V) units may be incrementally added or removed on an "as needed" basis.

Distributed processing also provides operational redundancy. Individual AN/GYQ-(V) units are paired and programmed, such, that either unit will perform the tasks of both in the event of a unit failure. In this mode, the system is neither interrupted nor is data lost.

Within component commands, AN/GYQ-21(V) Systems process and analyze locally collected intelligence data. The systems maintain local data bases and can transfer to and access the data bases of other AN/GYQ-(V) Systems at local, regional and national levels, as required.

Units are also being used for local and regional data processing, for "front-ending" host computer systems, and as a communication controller/ message handler for worldwide communication systems.

NMIC and PDSC are examples of military intelligence systems that include multiple 21(V)s operating in a functionally distributed multiprocessor mode.

3.1.5 AN/GYQ-21(V) Restrictions/Limitations

The purpose of this section is to present those architectural, physical and operational restrictions or limitations which will have to be observed during the design of 21(V) monitor. While not all 21(V) installations have the same set of constraints, the ones presented herein all occur with sufficient frequency to be important.

3.1.5.1 Architectural Considerations

The processors mostly found in 21(V) systems are the PDP-11/45 and the PDP-11/70. In this section we will discuss those differences between these processors, already alluded to in Section 3.2, that will have an impact on the monitor design. The PDP-11/45 and 11/70 (Figure 3-3) are both medium scale general purpose computers, designed around the basic PDP-11 family architecture. Although both are 16-bit machines, the 11/70 employs the power of a cache memory, a 32-bit word organized memory and I/O structures to demanding, multi-functioned computing requirements. This in conjunction with the UNIBUS Mapping Register and a unique Memory Management Technique are the major differences between the 11/45 and the 11/70 systems.

The PDP-11/45 memory management unit is designed for systems with memory sizes greater than 56K bytes and multi-user, multi-programming requirements which dictate memory protection and relocation. In this environment several user programs are run simultaneously. This is accomplished in memory management registers which generate an 18-bit physical address from the 16-bit virtual address produced by the processor.

The 11/70, however, also uses memory management to generate a 22-bit address from the virtual address to be placed in cache and main memory.

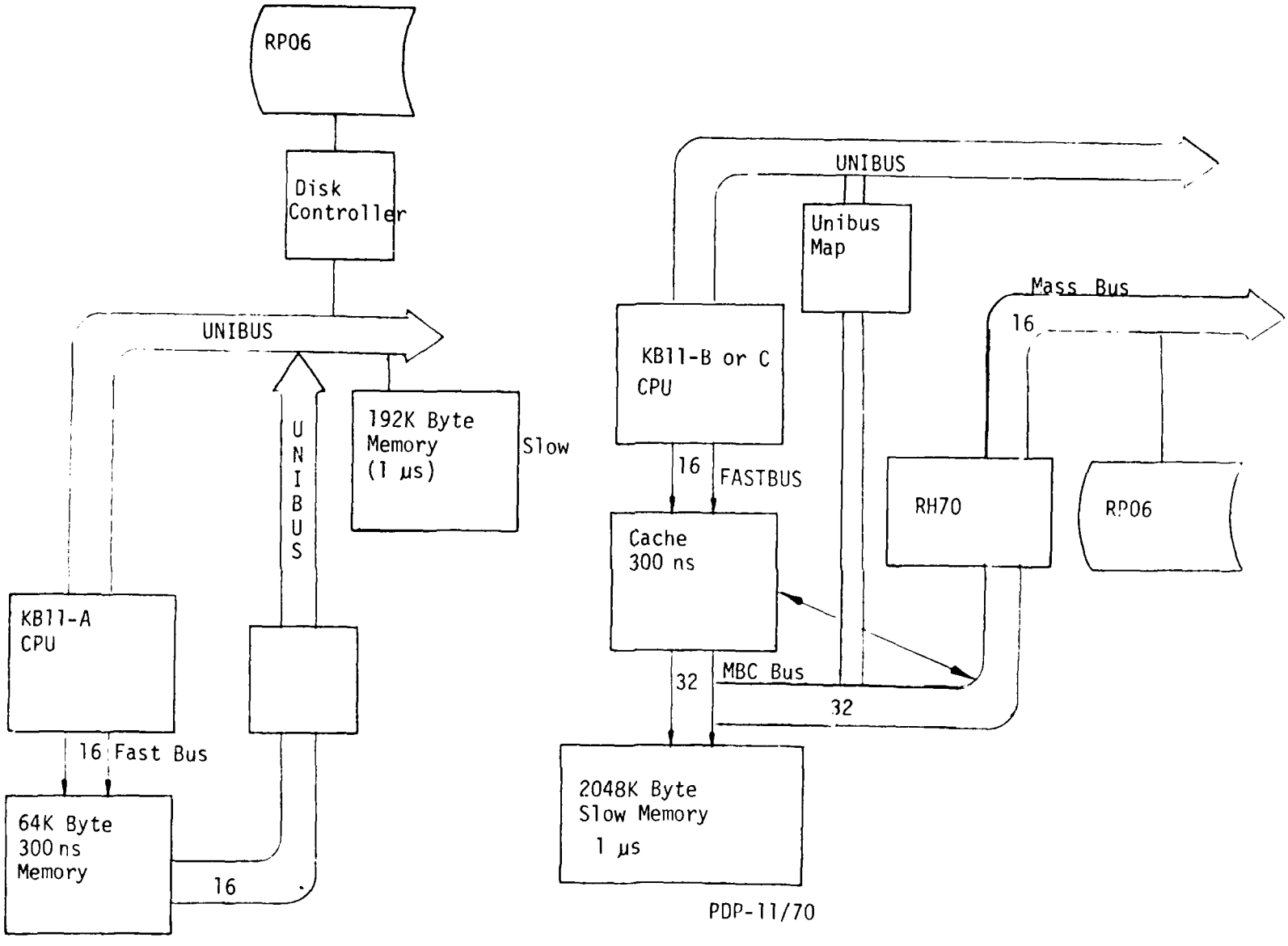


Figure 3-3 PDP-11/70 vs. PDP-11/45

The cache used in the 11/70 is two-way set-associative, consisting of two groups of 256 blocks each. Each block consists of two words, thus providing storage space for 1K words. The cache is implemented using a random replacement strategy with write-through.

The UNIBUS interface is the same for either the 11/45 or the 11/70. The Front Panel connections for the 11/70 are different from the 11/45, but are functionally similar. As a result a different cable would be required with a 11/70 vs. a 11/45. Any special individual probe points would be different for the 11/70 and 11/45, but, considering only a few would be necessary and they would only have to be installed once, this should not cause a problem.

3.1.5.2 Physical Constraints

The major physical constraints in interfacing with a 21(V) system, concern space, temperature, signal noise, load sensitivity and power.

- o Space

  Space for the main body as well as probe/connect elements of the hardware monitor is very limited in the majority of 21(V) installations: this is true both in the immediate vicinity of as well as inside the various 21(V) equipment enclosures. High card profiles, limited inter-card space and existing cable runs will make access to on-card signal points difficult without resorting to the use of card extenders.

- Temperature

  Card extenders cannot be used because the 21(V) components cannot be operated for any period of time with cabinet doors open and drawers extended because of the loss of already limited cooling air, danger of physical damage to the exposed modules and compromising emanations. The mounting of monitor components and the routing of monitor cables must be such that no disruptions or redirection of cooling air flow occurs.

- Signal Sensitivity

  Monitor cables must be routed so as to preclude cross talk with 21(V) components. Monitor cables should not provide paths for the sympathetic conduction and subsequent radiation of 21(V) signals.

- Load Sensitivity

  The UNIBUS is an asynchronous bus with bus length and load a factor in bus throughput. The hardware monitor should not significantly extend the UNIBUS length. The UNIBUS, being a transmission line, is also sensitive to reflections due to excessive stubs. Hardware monitor UNIBUS probes must have active components to provide a suitable match with the bus.

- Power

  Adequate amounts of 21(V) component power are available for the use of the immediate probe/interface elements only. The main units of the monitor should have self-contained power supplies which operate on 48-63 Hz AC power.

3.1.5.3 Operational Limitations

Three factors in this area are paramount in establishing design objectives - security, operational availability, and monitor operational procedures.

- o Security.
  The presence of long probe leads in commercially available monitors could result in unacceptable radio emissions of computer data. Significant care had therefore been taken to minimize cable lengths and to provide adequate shielding if and when longer cables are necessary. Similarly, probes are designed to eliminate requirements for open bay doors and rack assemblies.

- o Operational Availability.
  The classical hardware monitor requires significant machine downtime - first to "open up" the equipment for probe placement; this could well be followed by a period of limited accessability or restricted operation because of the extra equipment, cable runs, repositioning of probes, and open drawers. Lastly, the equipment must be removed and the system is again buttoned up. Not the least of the potential problems are those associated with probe placement. Given documented cases of misplaced probes and component destruction from, for example, short circuited leads, the likelihood of actual machine failure should not be underestimated.

  Such downtime is not usually acceptable to the 21(V) user community. The monitor design therefore:

  - minimizes the possibility of probe misplacement

- provides mechanically secure latches for those probes that are required

- may be "permanently" attached at low cost

  -- fu[illegible]system operation (closed drawers

  -- little or no accessability problems

  -- minimal need to change probes (no need at all for standard measurements)

  -- minimal downtown for installation or removal

o Operational Procedures.

The lack of highly trained computer performance evaluation professionals within the staff of a typical operational installation need not be a barrier to regular collection and reporting of measurement data for later evaluation. The most important criterion to accomplish the collection is to make the collection procedures a standard part of normal responsibilities that can be executed by the typical machine operator.

This, in turn, requires the monitor to be simply controlled and "programmed" through regular operator console devices (switch panels, teletypes, video terminals, etc.) - including the 21(V) console itself.

Given simplicity and relevancy of measurement, it is then possible (and cost effective) to consider permanent installation of a monitor configuration. Both operational and technical advantages rapidly accrue:

- o System "troubleshooting" can be better supported by remote experts.
- o A standard measurement set collected during daily operations will help system managers better understand the impact of operational procedures on system loading and responsiveness and to better predict performance trends.
- o More complex evaluations could be performed by specialists without the obvious need to install any new probes or hardware on the 21(V). The evaluations could be accomplished in support of system balancing, software tuning, or long range configuration planning.
- o System-wide (multiple installations) tests could be designed and executed by just transmitting (electrically or physically) test instructions to the installations and analyzing returned reports and tapes.
- o The sophisticated high speed event detection logic of the monitor can aid in the analysis of system failures and in the isolation of faulty elements (both hardware and software).

Improvement to a system MTBF is expected to result; "flaky" or intermittent gear might be rapidly isolated before a full failure brings down the system. MTTR would also be significantly improved as the failure characteristics are already known.

The goal of simplicity also has relevance for several technical issues including:

- Monitor cost
- Monitor measurement flexibility and completeness
- Independence of monitor and host software/hardware.

A final key element of the approach taken is the minimization of expensive display, printing, and bulk memory devices. This is accomplished at two levels:

- Many measurements can be stored within the monitor RAM and reported by simple LED/LCD displays.
- For those measurements requiring bulk memory and more extensive reporting, the 21(V) peripherals will be utilized whenever possible.

If the host 21(V) peripherals are fully utilized or there are other operational requirements precluding their use by the monitor, only those devices needed for the measurement process and unavailable on the host would be added to the monitor configuration.

3.2 AN/GYQ-21(V) Monitor Probe Point Locations

Five major types of probe sources have been identified for the AN/GYQ-21(V). These are:

- o UNIBUS Interface
- o Console/CPU Interface
- o Maintenance Interface
- o Discrete Points
- o Memory/Cache Interface

The monitor's probe point connections are shown in Figure 3-4. It consists of a probe board that would plug into an UNIBUS slot, and a pair of ribbon cables that would connect it to the Front Panel and Maintenance Interface Board. In addition certain critical signals are wire-wrapped to the probe board to provide access to these signals that otherwise cannot be obtained. Table 3-1 summarizes the individual signals which are required as inputs to the Hardware Monitor.

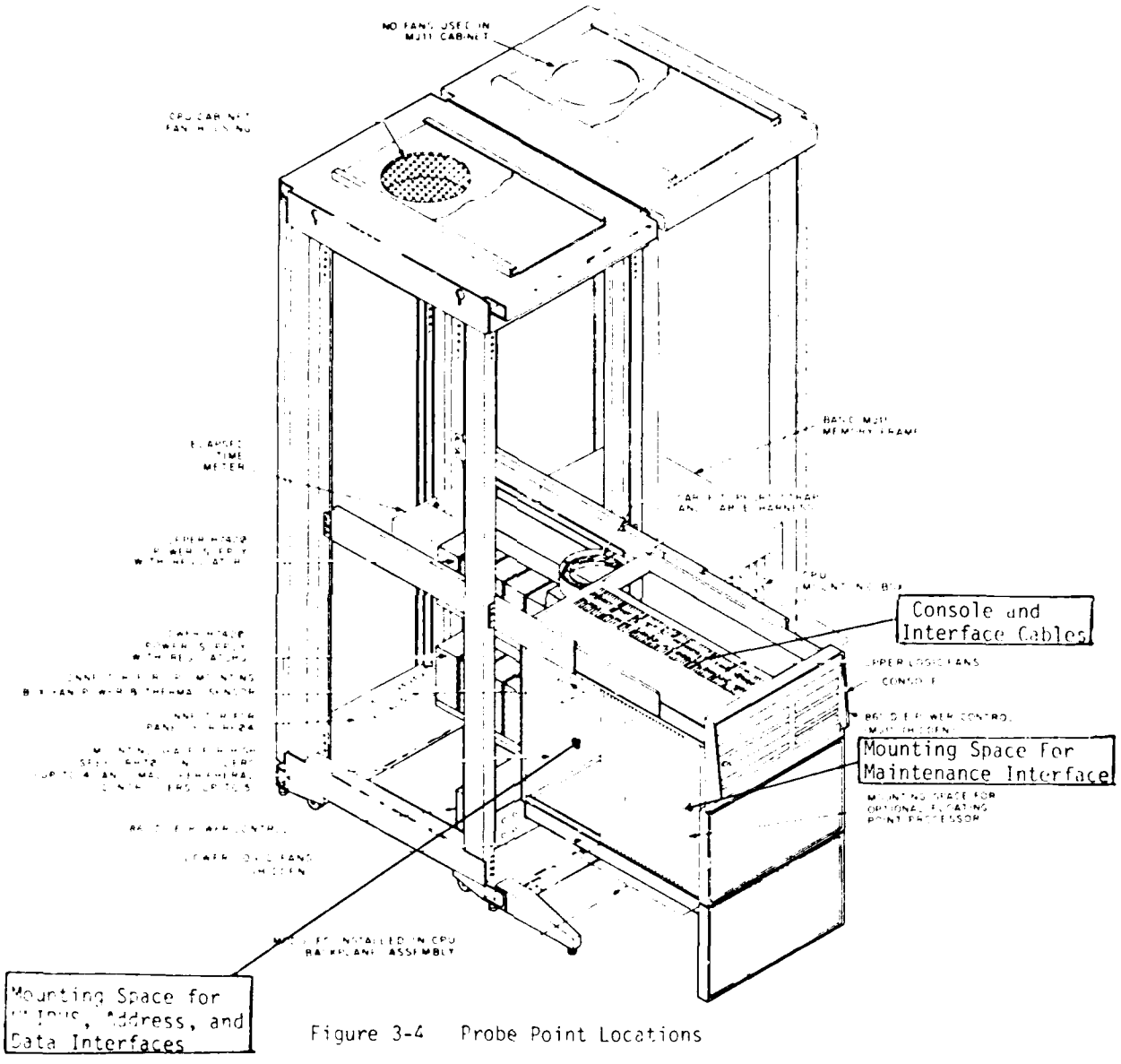


Figure 3-4 Probe Point Locations

GENERIC SIGNAL	NUMBER OF LINES		INPUT INTERFACE
Address	22 max.	o	Unibus
		o	Console/CPU
		o	Memory/Cache*
Data	16 max.	o	Unibus
		o	Console/CPU
		o	Memory/Cache*
Unibus Control	20	o	Unibus
CPU Control	17	o	Console/CPU
CPU Status	8	o	Maintenance
		o	Discrete
Memory Control	7	o	Maintenance
		o	Discrete
Floating Point Control	4	o	Maintenance
Timing and Control	10	o	Maintenance
MBC Control	8	o	Discrete*
	112 total		

*PDP-11/70 only

Table 3-1 Summary of Probe Points

3.2.1 Interfaces Between AN/GYQ-21(V) and Hardware Monitor

3.2.1.1 UNIBUS Interface

The UNIBUS Sensor Module interfaces directly with the 21(V) UNIBUS. In this way access is achieved to all UNIBUS activity (except for the grant lines) without affecting UNIBUS performance in any way. The Hardware Monitor will reside as an active peripheral device on the UNIBUS. Its location on the UNIBUS will be determined in part by the physical characteristics and design of the device itself, i.e., it may be designed to fit into an RH70 location or a Small Peripheral Controller (SPC) slot, or it may be designed as an independent unit with UNIBUS cables connecting it to the rest of the system.

The UNIBUS Interface (Tables 3-2, 3-3) provides 56 active signals to the Hardware Monitor, of which 54 are utilized by peripheral devices. It should also be noted that both the address and data signals are available via the Console/CPU Interface.

SIGNAL NAME	LOCATION	REQUIRED	COMMENTS
A00-A17	Refer to Table 3-3	yes*	Unibus address lines 00-17
ACLO		yes	System power failure
BBSY		yes	Unibus busy
BG4-7		yes	Bus grant at priority "n"
BR4-7		yes	Bus request at priority "n"
C0, C1		yes	Determine type of cycle
D00-D15		yes*	Unibus data lines 00-15
INIT		yes	System initialization (reset)
INTR		yes	Interrupt
MSYN		yes	Master sync
NPG		yes	Non-processor grant
NPR		yes	Non-processor request
PA, PB		no	Parity error; available but not utilized
SACK		yes	Selection acknowledged
DCLO		yes	System power failure
SSYN		yes	Slave sync

* Also available via Console/CPU Interface

Table 3-2 UNIBUS Interface

SIGNAL	PIN	SIGNAL	PIN
A00 L	BH2	D06 L	AF1
A01 L	BH1	D07 L	AH2
A02 L	BJ2	D08 L	AH1
A03 L	BJ1	D09 L	AJ2
A04 L	BK2	D10 L	AJ1
A05 L	BK1	D11 L	AK2
A06 L	BL2	D12 L	AK1
A07 L	BL1	D13 L	AL2
A08 L	BM2	D14 L	AL1
A09 L	BM1	D15 L	AM2
A10 L	BN2	GROUND	AB2
A11 L	BN1	GROUND	AC2
A12 L	BP2	GROUND	AN1
A13 L	BP1	GROUND	AP1
A14 L	BR2	GROUND	AR1
A15 L	BR1	GROUND	AS1
A16 L	BS2	GROUND	AT1
A17 L	BS1	GROUND	AV2
ACLO L	BF1	GROUND	BB2
BBSY L	AP2	GROUND	BC2
BG4 H	BE2	GROUND	BD1
BG5 H	BB1	GROUND	BE1
BG6 H	BA1	GROUND	BT1
BG7 H	AV1	GROUND	BV2
BR4 L	BD2	INIT L	AA1
BR5 L	BC1	INTR L	AB1
BR6 L	AU2	MSYN L	BV1
BR7 L	AT2	NPG H	AU1
C0 L	BU2	NPR L	AS2
C1 L	BT2	PA L	AM1
D00 L	AC1	PB L	AN2
D01 L	AD2	+5V*	AA2
D02 L	AD1	+5V*	BA2
D03 L	AE2	SACK !	AR2
D04 L	AE1	DCLO L	BF2
D05 L	AF2	SSYN L	BU1

* +5V IS WIRED TO THESE PINS TO SUPPLY POWER TO THE BUS TERMINATOR ONLY.

* +5V SHOULD NEVER BE CONNECTED VIA THE UNIBUS BETWEEN SYSTEM UNITS.

Table 3-3 UNIBUS Pin Assignments (By Signal Name)

3.2.1.2 Console/CPU Interface

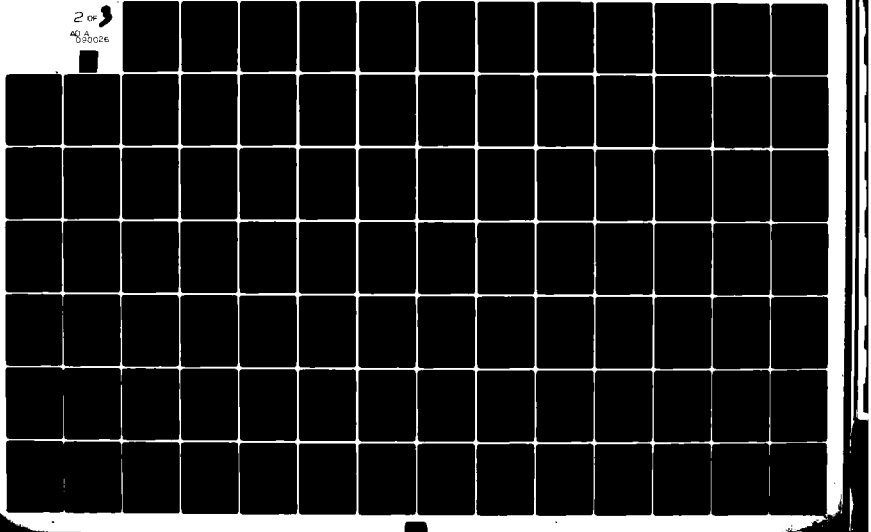
The console status/control module provides the basic monitor with access to the status indications provided on the 21(V) CPU console. It also provides the basic monitor with the ability to drive selected system control lines which emanate from the CPU console switches.

The Front Panel board gets its data through ribbon cables from the Processor Data and UNIBUS register board (PDR, slot 10, 11/70) and the SYS DESC/CNSL cables board (SCC/CNSL slot 16, 11/70). By inserting a special connector at this point, the Front Panel can still be connected, but now a cable with all of the Front Panel signals is available. A great deal of information can be obtained from just this connection. In fact, Digital Equipment Corporation's own hardware monitor (DIAMOND) connects through the Front Panel and CPU boards only.

DIAMOND was designed almost exclusively for software optimization. It has been used on VAX and PDP-11/34, 70 for development of RSX-11M+. The purpose of the DIAMOND monitor is to capture, in real time, percent CPU utilization, I/O times, I/O-CPU overlap, percent CPU in kernel mode, context-switch time, histogram of disk statistics. It actually consists of a PDP-11/04 connected through "spaghetti" to the CPU processor board and Front Panel (CPU Front Panel is removed.). A high level language was used to generate microcode for use by the 11/04 in looking at proper signals.

Design of both the PDP-11/45 and 11/70 make many signals available for visual display on the CPU console. These signals can be used by the Hardware Monitor in conjunction with the console when special cables allow connection to the Hardware Monitor. Ideally, the special cables will simply add another set of flat ribbon cables to the

AD-A090 026 MEASUREMENT CONCEPT CORP ROME NY F/G 9/2
MINICOMPUTER HARDWARE MONITOR DESIGN.(U)
JUN 80 B MORITZ, H SPAANENBURG, A J LABOUT F30602-79-C-0006
UNCLASSIFIED RADC-TR-80-203 NL



existing set, but major differences between the 11/45 and 11/70 may make this difficult to achieve. However, it should be possible to use the same Hardware Monitor connection scheme for both processors, either through patch plugs in the Hardware Monitor or by using a different type of cable arrangement, so that inputs to the Monitor are mated to the corresponding console signal.

There are four connectors on the Front Panel board in the 11/70, and three in the 11/45. The signals on these connectors are listed in Tables 3-4 and 3-5. The signals at this point are normally either high level lamp driver inputs or static manual switch type outputs. Hence, the risk of disruptive affects on CPU operation due to the presence of this interface is negligible. The signals of interest are all TTL level signals and all of these signals have only one TTL load on them. This means that standard TTL low power Schottky buffer circuits can be used in the monitor connection to the Front Panel board without concern for overloading the driving circuits.

The Address Data Path select switch on the Front Panel determines what address mode is displayed (sent to) on the Front Panel address lights. It is desirable to have the monitor change the switch setting during monitoring (dynamically) if possible. The select signals (DISP DATA SEL 0 and 1) go through J1 SS and TT. Thus by blocking these pins from the Front Panel and supplying these signals from the monitor, the switch setting can be set dynamically by the monitor.

SIGNAL NAME	LOCATION	REQUIRED	COMMENTS
DISP D00-15	J1	YES	Data (16 bits)
DISP DATA SEL0 DISP DATA SEL1	J1 J1	YES YES	Allow HM to select data source • FPP & CPU Microaddress • light register (display) • shifter (data paths) • bus register
VA00-03 DISP ADRS04-21	J2	YES	Address (22 bits)
DISP ADRS SEL0 DISP ADRS SEL1 DISP ADRS SEL2	J2 J2 J2	YES YES YES	Allow HM to select address source • program physical • kernel D virtual • kernel I virtual • console physical • supervisor D virtual • supervisor I virtual • user D virtual • user I virtual
IND PAUSE	J2	YES	CPU is in a PAUSE condition
IND MASTER	J2	YES	CPU is Unibus master
IND RUN	J2	YES	CPU is in RUN mode
IND ADRS ERR	J2	YES	Error when addressing Unibus device or memory
MMR0 MODE0 MMR0 MODE1	J2	YES	Used to decode user mode (K, S, or U)
SWR00-21	J3	NO	Switch Register (22 bits)

Table 3-4 PDP-11/45 Console/CPU Interface

SIGNAL NAME	LOCATION	REQUIRED	COMMENTS
DISP PAR HI DISP PAR LO	J1 J1	NO NO	Indicate parity of memory data
DISP D00-15	J1	YES	Data (16 bits)
DISP DATA SEL0 DISP DATA SEL1	J1 J1	YES YES	Allow HM to select data source • FPP & CPU Microaddress • light register (display) • shifter (data paths) • bus register
VA00-03 DISP ADRS04-21	J2	YES	Address (22 bits)
DISP ADRS SEL0 DISP ADRS SEL1 DISP ADRS SEL2	J2 J2 J2	YES YES YES	Allow HM to select address source • program physical • kernel D virtual • kernel I virtual • console physical • supervisor D virtual • supervisor I virtual • user D virtual • user I virtual
IND PAUSE	J2	YES	CPU is in a PAUSE condition
IND MASTER	J2	YES	CPU is Unibus master
IND RUN	J2	YES	CPU is in RUN mode
IND ADRS ERR	J2	YES	Error when addressing Unibus device or memory
IND PAR ERR	J2	YES	Parity error when accessing memory
MMR0 MODE0 MMR0 MODE1	J2	YES	Used to decode user mode (K, S, or U)
SWR00-21	J3	NO	Switch Register (22 bits)
IND 16 BIT MAPPING	J3	YES	Virtual address
IND 18 BIT MAPPING	J3	YES	Unibus address
IND 22 BIT MAPPING	J3	YES	Physical address

Table 3-5 PDP-11/70 Console/CPU Interface

3.2.1.3 Maintenance Interface

Both the PDP-11/45 and 11/70 make certain critical signals available for maintenance purposes. Originally provided for use with a CPU/FPP Maintenance Module, these signals can be made available to the Hardware Monitor through either a single-height or double-height paddle board. The paddle board terminates one end of the cable carrying signals between the maintenance slot and the Hardware Monitor. Table 3-6 lists the signals that are available at the maintenance interface.

3.2.1.4 Discrete Point Modules

The addition of discrete point sensor module probes will provide additional data not obtainable from the UNIBUS, Maintenance Interface and Front Panel alone. This group of signal probes is to be avoided if possible. However, it is clear that certain measures cannot be made without some of these connections.

This type of connection is not pleasing; however DEC PDP-11s are wire wrapped and a few back plane modifications are certainly possible at system integration time (for instance when assembling the 21(V) system).

The signals listed in Table 3-7 must be input to the Hardware Monitor by discrete signal lines running from pins on the backplane to the Hardware Monitor. It is possible for these discrete signal lines to be terminated on a paddle board (for convenience) before being routed to the Hardware Monitor. The relative worth of some of these signals may be of question. For instance, the signal UIRK is not really needed if the Monitor can determine the clocking of the Instruction Register by other means. Also, it is debatable whether eight signal lines are to be used just to gather information regarding MBC arbitration. Although the total number of discrete points listed is relatively small, this still represents the most desirable method of collecting data.

SIGNAL NAME	LOCATION*	REQUIRED	COMMENTS
MAT Z	D1	YES	These signals are the C, V,
MAT C	E1	YES	Z, N and T condition codes
MAT V	M1	YES	which make up bits 0-4 of
MAT N	U2	YES	the Processor Status Word.
PS04 MAT	P2	YES	
CONTROL OK	E2	YES	These signals control
SLOW CYCLE MAT	L1	YES	the memory cycles.
MEM SYNC	N1	YES	
BUST MAT	R2	NO	
FP ATTN	F1	YES	These signals control the
WAITS	S2	YES	Floating Point Processor/
FP REQ	T2	YES	CPU interaction.
FP SYNC MAT	V1	YES	
SSYN MAT	D2	NO	Same as corresponding
BBSY MAT	K1	NO	Unibus signals
MSYN B MAT	M2	NO	
AERF MAT	F2	NO	Address error, parity
PARITY ERR MAT	H2	NO	error, and stack limit error will all cause a
SERF MAT	L2	NO	TRAP or ABORT.
T1 MAT	H1	NO	These are outputs from the
T2 MAT	P1	NO	timing generator and may be
T3 MAT	R1	NO	used for timing and synchro-
T4 MAT	S1	NO	nization within the HM.
T5 MAT	N2	NO	
TPH MAT	K2	NO	
S1	B2	NO	These signals control the
S2	V2	NO	timing generator and can be
S3	A1	NO	used to stop the CPU in any
S4	U1	NO	predetermined state.

* Slot 1B in the PDP-11/70
  Slot 1F in the PDP-11/45

Table 3-6 Maintenance Interface

SIGNAL NAME	LOCATION	REQUIRED	COMMENTS
PS05 (1)	F10P2	YES	Processor Status Register
PS06 (1)	F10M1	YES	bits 5-7 allow tracking
PS07 (1)	F10K2	YES	of CPU priority.
UIRK	C8U2	YES	Instruction Register clock pulse.
UB REQUEST	D17R2	YES	These signals are used
MBC REQ	F17U2	YES	to determine which input
AMX S0	D17L1	YES	to the Cache Request Arbi-
AMX S1	D17K1	YES	trator is being honored.
CTRLA REQ	F21K1	YES	These signals are used
CTRLB REQ	F21K2	YES	to determine which MBC
CTRLC REQ	F21E2	YES	device is honored by the
CTRLD REQ	F21C1	YES	MBC Arbitrator.
SELADRS CTRLA	F21P2	YES	
SELADRS CTRLB	F21M2	YES	
SELADRS CTRLC	F21F1	YES	
SELADRS CTRLD	F21H2	YES	

Table 3-7 Discrete Points

The use of discrete probes to the various AN/GYQ-21(V) peripherals should be discouraged, although they would simplify the measurement procedures relative to a particular peripheral. Not only would these lines be long, they would be susceptible to disruptive changes when the set of peripherals change.

3.2.1.5 *Memory/Cache Interface*

The Memory Sensor Module provides access to the PDP-11/70 memory subsystem via the cache-unit where the memory operations from all target system devices are concentrated. Physical access is made at the cache end of the Memory Bus cable and at selected cache backplane signal points. While these signal points are moderately sensitive, when suitable isolation methods are adopted on the probe lines no deleterious effects due to monitoring should occur.

The signals that can be obtained by this interface are included in the listing of discrete probe points (Table 3-7).

In the PDP-11/70 it may be possible to utilize an RH70 location to house the logic assembly which will collect the event data and reduce it to a form which can be used by the Hardware Monitor Processor. The advantage is that Unibus signals plus physical address and data signals are available in the RH70 location. This would reduce the complexity of cabling in the Hardware Monitor system.

The advantages and disadvantages of various hardware configurations for the Monitor must be carefully weighed before a decision is made concerning the final hardware design.

3.2.2 Description of Interface Configuration

The hardware interface between the 21(V) and the Hardware Monitor will consist of a group of modules and cables, which allow the Hardware Monitor to collect and evaluate performance parameters of the host computer system. The specific configuration of these modules and cables will depend in part on the type of host computer and in part on the final design of the Hardware Monitor logic. Analysis of options concerning the microprocessor will affect its choice, and this may in turn impact the final Hardware Monitor configuration.

The Hardware Monitor will consist of four functional blocks:

- o data collection and reduction logic,
- o microprocessor,
- o microprocessor interface, and
- o PDP-11 interface.

For all Hardware Monitors the first three elements can be identical. Major differences between the PDP-11/45 and the PDP-11/70 dictate that the actual physical interface between the Hardware Monitor and the PDP-11 be unique.

The actual hardware configuration will depend on the type of microprocessor chosen for the Hardware Monitor and the particular PDP-11 host system. Figure 3-5 depicts the simplest Hardware Monitor configuration. Here, the entire Hardware Monitor, including microprocessor, is comprised of a set of boards which mount directly into an RH70 location in a PDP-11/70.

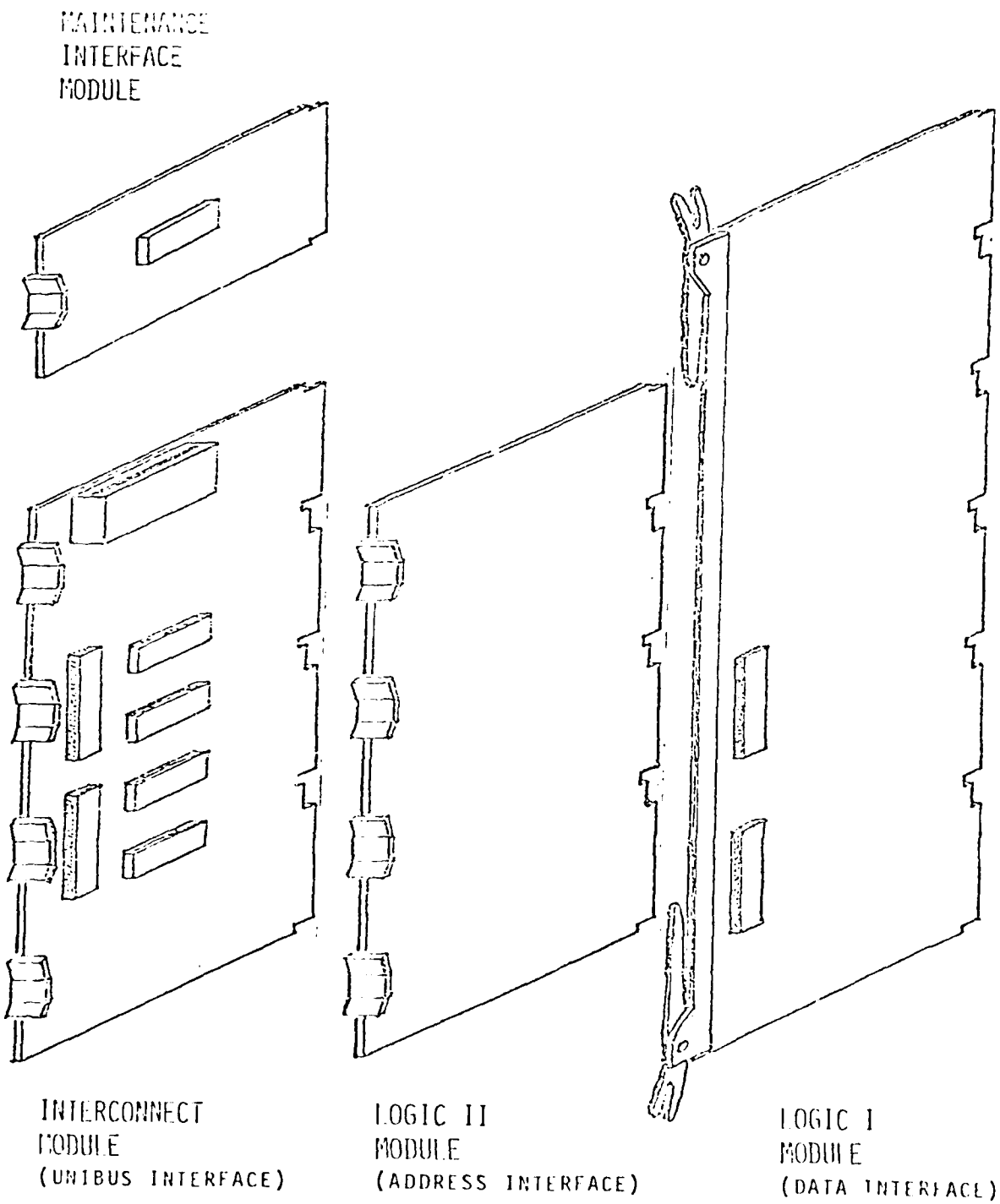


Figure 3-5 Hardware Monitor Configuration (Boards)

The Logic I and Logic II modules contain all Hardware Monitor logic functions. The Maintenance Interface and Interconnect modules provide the means for connecting PDP-11 signal inputs to the Hardware Monitor logic via ribbon cables between various boards.

If the microprocessor and associated circuitry is essentially a stand-alone system, then an interconnection can be made between the Hardware Monitor modules and a separately mounted microprocessor chassis.

In the case of the PDP-11/45 there is no appropriate location within the CPU for the installation of the Hardware Monitor modules of Figure 3-5. Furthermore, pin-outs for various UNIBUS signals are not the same between the RH70 "UNIBUS Interface" location and a SPC (Small Peripheral Controller) location. This implies that the Hardware Monitor Interconnect Module must be unique. It appears that for a PDP-11/45 system, the Hardware Monitor must reside in its own chassis. In this case the backplane of the chassis could perhaps reassign the various UNIBUS pin-outs so that only one set of boards would be required for the Hardware Monitor. It is also possible that a standard DD11 backplane could be modified for use as the Hardware Monitor backplane, and that it could be rewired to look like a RH70 backplane.

The electrical location of the Hardware Monitor will be critical in many system performance measurements which count or time certain UNIBUS signals. It should also be kept in mind that, as a peripheral device, the Hardware Monitor will also contribute one standard load to the system UNIBUS. Because of these factors, there will always be certain restrictions to the use and operation of the Hardware Monitor. Although we cannot expect 100% utility, we must formulate the operational standards to include as

many systems as possible. (The idea of making the Hardware Monitor a UNIBUS "repeater" may just add to the complexity of operational characteristics.)

The critical signals for UNIBUS location are the grant signals, BG and NPG. The ideal location for a probe for these signals is on the CPU output; then the probe is the first receptor. These signals are available at the UNIBUS terminator location (slot 1), but use here requires that a special terminator connector assembly be fabricated.

In the PDP-11/45 the grant signals go from the terminator directly to SPC slot #1. Thus, the Hardware Monitor could replace the device in slot #1, if this device can be relocated in the system. In the PDP-11/70 the grant signals take an irregular path after leaving the terminator. The NPG, BG7, BG5, and BG4 go to the SPC slots before going to the RH70 locations. The BG6 signal goes to the KW11-L, the RH70 slots, and then the SPC slots. If the Hardware Monitor is placed in RH70 location A, the NPG, BG7, BG5, and BG4 signals are first received by those controllers in the SPC slots. This is not critical for the NPG and BG5 signals, as they normally are not used by these controllers. Likewise, the BR7 signal is not critical because there are normally no BR7 devices in the SPC slots. The BR4, however, is used extensively by SPC devices, and some restriction may have to be placed on its use by the Hardware Monitor. (An alternative use of a discrete probe for this signal is certainly possible.)

Restrictions governing the use of a Hardware Monitor may become a touchy situation. We do not feel that a requirement of one UNIBUS load and one UNIBUS (or RH70) location is too extreme, considering the cost restrictions placed on the Hardware Monitor itself. It would be possible, with additional cost and complexity, to do away with the UNIBUS requirements, but this may be impractical.

3.2.3 Measurement Set and Probe Points

The Mc^2 hybrid monitor is capable of measuring and following the pulse of the 21(V) system. Many of the measurements are directly measured from the UNIBUS and Front Panel; however, with the vector generator, programmable patchboard and microprocessor it is possible to measure many events needing some processing before being useful to the user.

This section will describe system performance parameters, required inputs to the Hardware Monitor for measurement of the performance parameter, and additional comments concerning the parameter and its measurement. Many suggestions from users of the 21(V) system have contributed to the list and most of them have been studied and included. As the system is used, it is likely more and different types of measurements not already discussed will be suggested. In that light, the Mc^2 design will be a flexible one to accommodate these future requirements.

The list is divided into the following categories:

- Group A: CPU Performance
- Group B: UNIBUS Performance
- Group C: Peripheral Performance
- Group D: Program Performance
- Group E: Fault Detection/Isolation

3.2.3.1 CPU Performance

3.2.3.1.1 Task Time

This is a measurement of CPU time and wall clock time between task initiation and exit. The measurement is helpful in demonstrating the effect of priority and therefore facilitates tuning of the system.

The times will be collected under software control, as it would be extremely difficult to track global references (EXECUTIVE calls) made by the task. The probe "points" involved are the ADRS space of task. The monitor must look at adrs of instructions, as opposed to instruction itself. For more details one is referred to Section 4.3, where the host software monitor is discussed.

3.2.3.1.2 Event Time

An event is the occurrence of a predefined combination of signals and/or changes of signals. The implication is that any combination of signals, including addresses and data, can be determined to be an event. The event detection logic must be sophisticated enough to select the following events:

- Specific address - both memory and peripheral page.
- Specific range of addresses - both memory and peripheral page.
- Specific data (i.e., specific instruction).
- Specific data sequence.
- Specific signal (CPU, FPP, UNIBUS, Memory).
- Specific combination of signals.
- Specific sequence of signals.
- Time domain relationships of any of the above - both time per unit event and events per unit time.

3.2.3.1.3 Time in Kernel, Supervisor and User Mode

A majority of the CPU measurements involve separately accounting for these operations while the machine is in different states (Kernel/ Supervisor/User-mode; priority level). Instead of just counting instructions per second and/or memory references per second on an overall basis with a single counter, separate counters could be used to measure CPU activity in each of the desired states. The measurement therefore boils down to determining the state at which the computer is executing. The front panel/console interface provides the signals MMR0 mode 0 and mode 1 that information necessary to count the operation as belonging to Kernel, Supervisor, or User-mode.

3.2.3.1.4 Time at Priority Level

This is a measure of the time the CPU operates at a particular priority level. The monitor must decode PS bits 5-7 to determine priority level since eight levels are possible. A multiple clock system must exist in the Hardware Monitor to measure percentage of time at various levels. Back plane locations F10P2, F10M1, F10K2, are processor status register bits 5 thru 7. These bits can be fed to the state vector to enable the counting of events at different CPU priorities.

The alternate to measuring these discrete points involves additional sophistication in tracking trap instructions, processor error traps and INTR traps - i.e., all those events that can cause a change in processor priority. The procedure makes use of the fact that upon any of the above traps, the CPU places the current PC and PSW on the R6 stack in memory and retrieves the new PC and PSW from the vector locations in low memory specified by instruction, error trap type or INTR sequence. As each transfer to/from memory by the processor is made available to the monitor via the state vector, the straightforward sequence control can pick out the PSW data that is to be loaded to the CPU's register.

The RTI instruction will also have to be monitored as it belongs to the trap class.

3.2.3.1.5 CPU Idle

This is a measurement of the time that the CPU is not busy, or better defined, the time that it is in the WAIT state. Two methods for measuring the occurrence of the WAIT state are available:

- Through monitoring the CPU micro address (available via the console). In the WAIT state the micro address cycles in states 11, 264 and 273 until an INTR puts it in state 244.
- By comparing fetched instructions to the bit configuration of a WAIT instruction. The Hardware Monitor can determine when a WAIT instruction is executed by looking at the data loaded into the instruction register. This is done by looking at the data in the Bus Register after IR loading time. Br and IR are loaded with the same data at the same time. The WAIT state concludes with an interrupt.

The latter appears more reasonable. For Unibus machines, the state vector will contain the fact that the transaction was a CPU fetch. The data lines in the state vector can be matched by the patchboard and cause an interrupt condition to the microprocessor.

For Cache machines, the only mechanism to provide equivalent sampling will be the use of the Front Panel data lines selected to show actual Unibus data. The match to the WAIT instruction would be identical as the state vector would now contain data lines from the Front Panel instead of from the Unibus. INTR is specifically a Unibus function so there would be no difference in the stopping of the WAIT time function.

3.2.3.1.6 CPU - I/O Overlap

This is a measure of the time that I/O is taking place while the CPU is executing. There is a close correlation between this measurement and that of UNIBUS utilization (especially for the 11/45), so both will be discussed at this time.

Maximum system throughput will occur if each and every device on the system is operating at its maximum rate. Two factors become intuitively obvious in evaluation of a particular system, sub-system or device:

- o If a device is ready to perform but is not used, it is not operating at its maximum rate, and
- o If a device must wait for another device to complete an operation before it can continue, it is not operating at its maximum rate.

The first factor can be considered device utilization, and the second factor can be considered device efficiency. We maximize system throughput by maximizing both utilization and efficiency. Total Unibus utilization = $\frac{t_{BBSY}}{T}$ i.e., the time BBSY is asserted over a particular period of time. CPU Unibus utilization = $\frac{t_{MASTER}}{T}$ i.e., the CPU is using the Unibus when the MASTER signal is asserted. CPU Unibus efficiency = $1 - \frac{t_{PAUSE} - t_{MASTER}}{t_{PAUSE}}$. The total time that CPU wants to use the Unibus is represented by t_{PAUSE}. It should be noted that t_{PAUSE} also includes the time that the CPU is waiting for a device to interrupt after a bus grant has been issued. This time may or may not be considered

a factor in CPU efficiency. I/O Unibus utilization = $\frac{t_{BBSY} - t_{MASTER}}{T}$

I/O Unibus efficiency = $\frac{t_{SACK}}{(t_{BBSY} - t_{MASTER}) + t_{SACK}}$. As with CPU Unibus efficiency, the I/O Unibus efficiency is maximum efficiency minus the time spent waiting for the bus to become available. CPU-I/O overlap should express itself in terms of Unibus utilization and efficiency. The BBSY and SACK signals are available at the UNIBUS, while the IND PAUSE and IND MASTER signals are available at the Console/ CPU interface. CPU-I/O Overlap in the 11/70 is more importantly a function of cache/memory performance than Unibus performance and will be treated in the section concerning peripheral performance (Section 3.2.3.3). For those devices utilizing the Unibus for data transfers, performance measurements for the Unibus are the same as discussed before.

3.2.3.1.7 CPU-FPP Overlap

This is a measure of the time that the FPP is operating at the same time that the CPU continues its instruction sequence. It indicates increased performance and speed obtained by applying the FPP. Signals that the FPP and CPU are in operation can be found on the Maintenance Interface.

3.2.3.2 Unibus Performance

3.2.3.2.1 Unibus Acquisition Time

Unibus acquisition time represents the time between assertion of a bus request (BRn or NPR) and the device's assertion of BBSY as bus master. Bus acquisition time is a complex measurement. Whereas NPG response to an NPR may be relatively quick, the UNIBUS "daisy chain" NPG architecture does not guarantee that the first requesting device receives the first grant. The BR/BG cycle can be even more complex as, in addition to the BG

"daisy chain" architecture, BGs will not be issued if the CPU priority level (Processor Status Word) is equal to or greater than the level at which a request is made. Therefore, execution of interrupt service routines (ISRs) or other high priority (hardware level) code will preclude the issue of BGs for times on the order of 100-150 microseconds.

A primary issue is the degree of detail necessary to adequately describe latency effects. Clearly, if each peripheral's NPR, BR, and BBSY lines (in front of bus transceiver) are probed, it is possible to answer questions such as:

- How long did this device wait for an NPG?
- How long did this device wait for a BR?

To get this level of detail, one must provide discrete probes for each device controller.

The alternative to this approach explores the statistical nature of system performance on a given BR/NPR line and correlates this information with device activity counts (number of NPRs, number of INTRs by device) made available through the state vector.

NPR and BR latencies are considered two different problems as an NPG will always be issued within one CPU instruction time whereas the BR could remain unanswered for 100 times that duration.

All signals for this measurement are available on the Unibus Interface.

3.2.3.2.2 UNIBUS Occupancy

As BBSY is always asserted by the current bus master, the measurement of BBSY AND "device active" determines the percentage of "busy time" used by each of the UNIBUS devices which may become bus master.

Measuring the time the CPU is master is extremely simple as the IND MASTER signal from the Front Panel provides the necessary data. Non-CPU devices are a little more difficult as they may become bus master for NPR transfers or at INTR time.

The identification of an interrupting device is relatively straightforward as it places a predetermined vector address on the UNIBUS at INTR time. The device performing an NPR can be most easily identified through discrete sensor lines attached, for example, to the input to the device controller's BBSY line UNIBUS transceiver.

However, this project continues to be guided by the desire to minimize or remove discrete probes except under those cases where such a discrete probe is permanently connected and independent of the peripheral device configuration.

The alternative to discrete probes is more complex but achievable with the suggested design. It requires that the monitor be "sensitized" to CPU loading of selected NPR device buffer address and word count registers. As the address to (from) which an NPR transfer is made appears on the UNIBUS and is on the state vector, comparison of such an address with the memory buffer bounds obtained from the register loading provides the required device identification.

Initially, a comparator buffer in the patchboard is set to the desired device(s) buffer address register. Upon receiving a state vector whose address lines match this value, the Hardware Monitor strobes the data lines of the vector (containing the buffer address) into a second comparator buffer.

If all buffer sizes are assumed to be multiples of 256 bytes, only the high order address bits are necessary. Otherwise a third comparator buffer is loaded with the sum of the buffer base address and word count (complemented as necessary).

To keep the patchboard logic simple, it is likely that the monitor microprocessor will actually perform the loading of the comparator buffers in response to interrupts generated by the matching of the device buffer address register on the patchboard.

As each state vector is received at the monitor, the address lines are examined. If the transfer is NPR AND the address is within the selected buffer, it is known that the selected device has been bus master. The MSYN-SSYN time will be close to the BBSY duration and can be "posted" to the time accumulator for the appropriate device.

A similar approach using the D lines of the state vector at INTR time can be taken. Here, a comparator buffer contains the selected device's interrupt vector instead of (or in addition to) its memory buffer boundaries. If the vector is an INTR event AND the D lines match the selected device's interrupt address, the event time is added to the appropriate time accumulator.

All signals for this measurement are available on the Unibus Interface (except IND MASTER, which is available on the Console/CPU Interface).

3.2.3.2.3 NPR Latency

Under most cases, a DMA peripheral that wishes to perform a transfer will issue its NPR and, upon receipt of an NPG will perform an MSYN/SSYN transfer cycle and relinquish bus master. It is possible (particularly with disks) for the controller, when bus master, to perform two MSYN/SSYN transfer cycles before relinquishing bus master. Upon asserting NPR, a device on the bus will wait until all devices closer to the CPU also having their NPR asserted are serviced with NPGs (and resultant transfer cycles).

If a device is not "serviced" with an NPG within a certain time (dependent on device) its data buffer could be overwritten by new data so that a "missed data" error would occur.

It is believed that an important use of the monitor is to aid system "balancing" so that missed data errors will not occur. The issue is to determine that information necessary to perform this needed function. Clearly, each device can be individually monitored through discrete probes (the brute force method). This method tells how long a given device typically waits (average, median, mode, variance) for its NPG.

However, it really doesn't matter how long it waits unless either the wait time exceeds the latency limit determined by the device's buffer size and transfer rate or, as in the case of a disk with "silo" (FIFO), if the average transfer rate during a transfer is well below that of the device itself. In the latter case, the determination can be made directly from the state vector and address matching technique discussed for the Unibus Occupancy measurement. The clock is started with the first transfer and stopped with the last transfer. This time, divided by the number of transfers, is the effective transfer rate of the device in presence of contention from other devices.

It remains to determine the probability that a given device will have to wait longer than a given amount of time for NPR bus access. First one must obtain the probability of NPR contention - that percent of all NPRs for which, when the NPG is issued, the NPR line remains high. It is only these transactions that, without discrete probes, we need to further explore in order to determine if NPR latency by device can be extracted.

Through the state vector address map, we already know the identities of the devices participating in NPR transfers at the time of contention and the order in which they are serviced by NPGs. We also know the physical bus locations at which each device is connected.

The simple contention involves two devices that both assert NPR within one bus handshake/transfer time - this "window" occurring during the duration of a single MSYN/SSYN preceeding the BG. The closer device will intercept the NPG and perform the NPG with a measured NPR/NPG time equal to that from the first assertion of NPR to the NPG. The second NPG will receive a NPR/NPG time equal to the time between the first assertion of NPR and the second NPG. In both cases, the NPR/NPG times are accurate to the extent that they show a latency of less than 1 handshake/transfer cycle and a latency of between 1 and 2 cycles, respectively. The inaccuracy lies in the inability to differentiate the first from the second requestor and is, in this case always less than one cycle.

It is believed that this degree of two device contention measurement accuracy is more than adequate for a monitor possessing the stated cost and environmental impact objectives.

Were three devices involved, the degree of uncertainty in the measurement would grow to between 1 and 2 cycle times. However, this worst case requires all three devices to assert NPR within two cycles such that the closest device asserts NPR during the same cycle of either of (or both of)

the others and that the third device asserts NPR before the closest device executes its grant. The error in timing NPR/NPG for the closest device is still less than 1 cycle.

As the very detection of a three-way contention and the measurement of its rate of occurrence and involved devices is believed more important than the precise latency by device, and as the effective throughput of each device is known anyway, a design without discrete probes at each NPR device is believed to be satisfactory.

3.2.3.2.4 BR Latency

Whereas excess BR latency does not normally lead to hardware errors such as missed data in 21(V) configurations, it can, in general, cause problems with input from non-DMA devices such as DL11s and will contribute to reduced system throughput.

Unlike NPR latencies which are measured in terms of a few bus cycles, BR latencies of several hundred microseconds are not expected to be unusual. Because of this longer time span, measurements without discrete device probes will suffer inaccuracies on the order of one or more Interrupt Service Routine (ISR) execution times. The probability of multiple BRs queued at a given level is much higher as, for example, no BGs for a lower level BR can be issued until all higher level BGs have been given.

The issue in this case involves the determination of need for BR/BG times by device - especially when the contending devices at a BR level are known as are the identities of all devices waiting at lower priorities and the net I/O times (device "GO" to device INTR). What is missing is an accurate determination of the fraction of net I/O time by device caused by BR queueing. Total system time during which BR queueing exists is, however, known by BR level through the contents of the state vector.

For cases where there are no multiple BRs at a given level, the current design does provide the BR queueing fraction of net I/O time by device: the total time of BRn assertion is known as is the identity of the device that uses the BGn for its interrupt.

Thus the issue boils down to the probability of multiple non-granted BRs at a given level and the importance of knowing the exact device BR/BG time in the fraction of cases where multiple BRs of a given level will be asserted. The worst case inaccuracies of the suggested (no discrete device probe) configuration are basically equal to the time between the assertion of the first BR at a given level and the issue of the first BG that de-asserts BR; this time measurement is provided by the suggested configuration.

3.2.3.2.5 INTR Latency

A measurement, similar to NPR and BR Latency used for device position/ priority evaluation. It measures the time between BR and INTR as found on the Unibus. All signals for INTR, BR and NPR latency can be found on the Unibus Interface.

3.2.3.2.6 UMR Utilization

The Unibus Mapping Registers extend the PDP-11/70 physical address available from 256K bytes to 4M bytes. The utilization is a measure of use of this feature. Usually the operating system does not dynamically allocate UMR usage. However, the address space of the mapping registers can be monitored on the Unibus Interface.

3.2.3.2.7 UNIBUS Effective Transfer Rate

This measurement is simply defined as the number of UNIBUS data transfers per unit of time (e.g., 10^6 transfers per second). It is a highly useful number on 21(V) machinery of the 11/45 class or below as it provides a baseline to which individual device activity on the same bus (also measured as transfers per second) can be compared. On a 11/70 class machine, the utility of this measurement depends entirely upon the number and type of peripherals using the UNIBUS.

The UNIBUS MSYN and SSYN signals together define a data transfer cycle on the UNIBUS and are preprocessed by the UNIBUS sensor module to create its state vector and time extent. Each state vector defines either a data transfer or a control transfer. Each arrival of a state vector describing a data transfer increments a data transfer cycle counter on the programmable patchboard. Counter overflow interrupts the monitor's microprocessor which obtains and stores the time at which overflow occurs. Assuming no more than 1×10^6 transfers per second and a 16-bit counter, the cycle counter will overflow each 33 milliseconds - thereby producing little microprocessor loading. Preloading the register will effectively shorten the time between interrupts and allow higher resolution.

3.2.3.2.8 UNIBUS Read/Write Count

The Unibus C0 and C1 signals determine direction (in or out), word size (word or byte out) and read-modify-write cycle. If the C0, C1 states are to be separately counted, a total of four counters are required - one each for three C0, C1 states and the fourth for the total number of cycles. Overflow of the total cycle counter causes the interrupted microprocessor to read each of the other three registers (and clear them) and to store this data with the time stamp for post processing.

3.2.3.3 Peripheral Performance

3.2.3.3.1 Memory References in Range

This measurement delivers a count or rate of memory references in a particular range of memory (e.g., a particular bank). The measurement can be further itemized by instructions only, CPU data only, read and/or write, by device or a combination of these categories.

3.2.3.3.2 Memory Range by Task

As for the Memory References in Range measurement the Hardware Monitor must be capable of selecting a window of addresses to be included in the determination of an event. Also included can be the C0, C1 information and whether or not the data represents an instruction. The signals for both these measurements come from the Unibus Interface, the Console/CPU Interface and the Memory/Cache Interface. The host software monitor will provide the Task Identification (see Section 4.3).

3.2.3.3.3 Controller Registers

Similar to the last two measurement. In this case, the window of addresses under consideration includes those of specific device registers. In particular, a "control" register can be selected and "GO" bit monitored. Of interest is a count by device of the number of "GO" events to the respective Control and Status Registers (CSR). Also measured is programmed I/O, that is when the CPU puts data on the lines for transfer without using NPR. Signals for this measurement are available on Unibus Interface and Console/CPU Interface.

3.2.3.3.4 Number of NPRs per Controller

This measurement is accomplished in a similar manner as Memory References in Range and Memory Range by Task. It requires separate event detection

logic for each controller to be dynamically evaluated. The device which asserted NPR is determined by monitoring the address lines during the MSYN - SSYN cycle. It could also be measured by software within any device handler. Again all signals for this measurement are present on the Unibus Interface and Console/CPU Interface.

3.2.3.3.5 Number of INTRs per Controller

The frequency of interrupts is determined by the INTR signal and the associated vector. The time of the Interrupt Service Routine can be measured as the time between the INTR signal and a subsequent RTI instruction. Interruption of the ISR itself can be detected by any intervening program break, which first appears when the CPU enters the BRQ service routine. All signals for this measurement are present on Unibus Interface and Console/CPU Interface.

3.2.3.3.6 Memory/Peripheral Transfers

In the PDP-11/45 memory transfers by peripherals are accomplished via NPR's. Total number of memory transfers can be determined by looking at the number of memory address space accesses.

In the PDP-11/70 memory accesses can be determined by looking at "request" signals in the cache control logic. This can be done either by backplane connection or by utilizing one of the RH70 locations. The pertinent signals are:

- o CONTROL OK - CPU memory reference
- o UB REQUEST - Unibus NPR memory reference
- o MBC REQ - Memory reference by RH70-type controller.

These signals are available on the Memory/Cache Interface and the Maintenance Interface.

3.2.3.3.7 Service Time

A measurement of the event to be accomplished and the duration of the INTR signal. The start of the event should be programmable to include the issuance of any command to any device. The event terminates with the INTR signal. It could also be measured by software within any device handler. All signals for this measurement are on the Unibus Interface and the Console/CPU Interface.

3.2.3.3.8 Number of Terminals On-Line

This information is tracked by the system EXECUTIVE. It can therefore be passed by the host software monitor to the hardware monitor microprocessor or it can be directly requested by the microprocessor.

3.2.3.3.9 Number of Units Busy

There are programs available for the host system to monitor this information. The way for the Hardware Monitor to track this information is by either having a discrete probe for each device or by having the capability of performing multiple Service Time evaluations.

3.2.3.3.10 Controller Busy

This is an extension of the service time measurement to include measurement of total service time per unit time.

3.2.3.3.11 Disk Performance

Each disk drive provides status signals to the controller to assist in performing the "handshake" functions. It might prove infeasible to attach a discrete probe to either the disk drive or the controller when this information can be provided by the controller status register. One of the most important parameters of disk operation is the "seek" time. This can be measured in the host system by having the disk handler issue "SEEK" commands followed by "READ/WRITE" commands rather than issuing "implied seek" R/W commands. The Hardware Monitor could measure the seek time by measuring the Service Time for a SEEK command. An extremely useful application of this measurement would be to dynamically restruction file organization on a disk pack to increase system efficiency. An application of this type may be beyond the scope of the Hardware Monitor system. Other parameters of the disk system are also available via the various status registers of the controller.

For additional disk performance measurements one is referred to Section 4.3 describing the host software monitor.

3.2.3.3.12 Cache Acquisition Latency

Latency is measured from the time a request is made to the time the Cache Request Arbitrator allows memory access. The request signals are listed under the Memory/Peripheral measurement. The output of the arbitration logic must be decoded to determine which request is serviced. The signals AMX SØ and AMX S1 are decoded as follows:

S0	S1	Cache Cycle
L	L	CPU
L	H	MBC
H	H	UNIBUS

The necessary signals for this measurement can be found on the Maintenance Interface and the Memory/Cache Interface.

3.2.3.3.13 Cache Hit Rate

First a clarification of terms is in order. A Cache hit refers to a memory read cycle where data is read from high-speed memory rather than slower core memory. Although intended primarily as a means of speeding up CPU memory references, the cache memory in the PDP-11/70 is also used during Unibus memory references, i.e., NPR transfers. The HIT/MISS Register associated with the cache memory only tracks CPU memory references, so it is not a true indication of the system cache hit rate.

The second point which should be mentioned is that any type of active sampling, such as reading the HIT/MISS REgister, should be avoided whenever possible to reduce artifacts. Fortunately, there is a very easy method of determining the cache hit rate without active sampling, and by selecting appropriate signals it is possible to obtain a better

picture of cache performance. The key signal to cache operation is called SLOW CYCLE. Whenever this signal is asserted by the cache control logic, the memory cycle requires reference to main memory. By proper decoding of the memory request signals, the C1 signal, and the SLOW CYCLE signal, it will be possible to determine the cache hit rate for any of the following performance measures:

- o all memory references
- o only READ memory references
- o CPU memory references
- o CPU READ memory references
- o Unibus memory references
- o Unibus READ memory references

The Maintenance Interface and the Memory/Cache Interface therefore contain the required signals for this measurement.

3.2.3.3.14 Cache/Memory Transfers

This is a simple extension of the Cache Hit Rate measurement. Where the Cache Hit Rate is determined by the number of hits per type of transfer, the Cache/Memory Transfer measurement is determined by the number of transfers (by type) per unit time.

3.2.3.3.15 MBC Busy

All system peripherals, including MBC-type devices, utilize the Unibus to communicate with the CPU. Thus, the MBC Busy measurement is identical to the Controller Busy measurement.

3.2.3.3.16 MBC Transfers

This is an extension of the Cache Hit Rate and Cache/Memory Transfers measurements. Although MBC transfers do not use cache memory, they are controlled by the cache control logic, and thus, the signals that are used to determine cache performance include those necessary to measure MBC transfers. This can be equated to transfers per unit time as in the Cache/Memory Transfers Measurement.

3.2.3.3.17 MBC Latency

Contention for MBC memory requests and thus the MBC Latency can be measured by accessing discrete signals on the backplane. All MBC controllers can make requests simultaneously; it is up to the MBC arbitration logic to select the controller with the highest priority. Latency is measured from request time (CTRL "X" REQ) to selection time (SELADRS CRTL "X"). These signals as well as signals from the Maintenance Interface are required for this measurement.

3.2.3.4 Program Performance

Measurement and evaluation of most Program Performance parameters are most effectively accomplished via the host system software. Exceptions to this where some data can be collected by the Hardware Monitor are noted.

3.2.3.4.1 Task Request Rate

Measurement performed by host system software.

3.2.3.4.2 I/O Request by Task/Device

Measurement performed by host system software.

3.2.3.4.3 Active Tasks in Memory

Measurement performed by host system software.

3.2.3.4.4 Memory Allocations

Measurement performed by host system software. Results will show memory fragmentation. Memory allocation can also be tracked via the Memory Management registers (PAR's and PDR's). This relatively easy to do, as the PAR's and PDR's occupy a contiguous block of addresses in the peripheral page.

3.2.3.4.5 Node Claim/Release Status

Measurement performed by host system software.

3.2.3.4.6 Instructions per Second

This information may be obtained in several different methods.

The first is a match to a micro address in the PDP-11/45 and PDP-11/70 processor. The micro address itself is available from the Front Panel with assertion of the proper multiplex signals. All PDP-11/45 and 11/70 processors use the ROM state at microaddress 343 to decode the current instruction. A count of the number of times the CPU goes to this microaddress is equivalent to the number of instructions fetched. This count can be made on a per unit time basis.

The second method is a discrete probe to the PDP-11 backplane where the signal C8U2 (IR clock) may be sampled. The IR is the processor's instruction register which is loaded with each instruction that the processor is to execute. The preferred method is the use of the IR load signal as (1) the PDP-11/45 and PDP-11/70 may share a micro address; this is not obviously the case for all processors in the 21(V)

line and (2) the IR load signal occurs when the instruction being fetched is on the data lines of the Unibus of fast bus. By using the IR load signal we are then able to capture the actual instruction being executed so that comparisons with selected instructions such as EMT or WAIT can be made.

3.2.3.4.7 Instruction Type

The instructions can be collected over a period of time and then be evaluated by the Hardware Monitor processor for operation code, mode, and register. Tne priority can be collected along with each instruction. Correlation between task and instruction will require a rather extensive exchange of parameters and status between the host processor and the Hardware Monitor processor. Again signals from the Console/CPU Interface or the IR clock are required to detect instructions.

3.2.3.4.8 Floating Point Instructions

Floating point instructions can easily be detected by looking for the operation code $(17XXXX)_8$. This information can be represented as total count or count per unit time.

3.2.3.5 Fault Detection and Isolation

Most of the events in this section are detected by the host system - either through hardware or software. Detection through hardware will cause a subsequent TRAP or ABORT condition, which will vector the software to a corresponding location. These events can be easily tracked by host system software.

3.2.3.5.1 Slave Faults

Slave Faults can be detected by the lack of a SSYN response when MSYN is asserted. The Unibus address where the slave fault occurs can be detected by the Hardware Monitor. The signals in question are on the Unibus Interface.

3.2.3.5.2 Trace Trap

Trace Traps are detected by monitoring the T bit in the Processor Status Word. Data and status associated with the trap can be collected by the Hardware Monitor from the Console/CPU Interface and the Maintenance Interface.

3.2.3.5.3 Error Counts

Host system software can collect pertinent error information via a task called ERRLOG. This program has been developed to collect parameters for mass-storage-type devices, such as disks and magnetic tape drives. Error information can also be collected by monitoring selective device addresses during an ISR (see peripheral performance measurements of Controller Registers, number of INTRs per Controller and Service Time). Interpretation of device registers can be accomplished by the Hardware Monitor processor.

3.2.3.5.4 Unclaimed NPG, BG

An unclaimed NPG or BG will cause a SACK timeout in the CPU. This will light an error indicator in the CPU but will not cause a hardware trap. This condition can be detected by the lack of the SACK response to either a BG or a NPG. In order to determine which device fails to issue the SACK response, a separate probe is required for each device, especially if the condition is intermittent.

3.2.3.5.5 Memory Out of Bounds

Host system software reports this condition, if the address is asserted during a CPU cycle. Device handlers will receive a non-existent memory error if there is no SSYN response after the device asserts MSYN.

3.2.3.5.6 Odd Address Error

Host system software reports this condition (similar to Memory Out of Bounds measurement).

3.2.3.5.7 Multiple NPRs/BRs

The occurrence of multiple NPR's or BR's is not a system error condition. It can occur anytime more than one device contends for use of the bus. The measurement for Multiple NPR's/BR's is simply the condition where the NPR or BR is not negated after SACK is asserted. This measurement is done using the Unibus Interface signals.

3.2.3.5.8 EMT Service Time

EMT timing is the time from issue of EMT by task to return of control by EXECUTIVE to the same or another application task. EMT and TRAP instructions can be detected by monitoring instructions for operation code $(104XXX)_8$. Number of instructions and number per unit time can be easily be evaluated. Service time begins with an EMT or TRAP instruction and is completed with a RTT or RTI instruction.

3.2.3.5.9 Event Driven Trap

This measurement is a subset of the EMT Service Time measurement. A particular event can be detected by the contents of the lower byte of an EMT or TRAP instruction.

3.2.3.5.10 Glitches on Power Supply

A discrete probe is required for each voltage within every power supply to be monitored. This could require many probes, for in the CPU alone there are at least eleven (11) different supplies. It is possible that one single power supply could be representative of all; this would require in-depth investigation.

3.2.3.5.11 Loss of Power

The ACLO and DCLO signals on the Unibus Interface are used by the system to determine loss of power.

3.3 Vector Generator

The 21(V) system will be measured by monitoring a number of probe points that will minimally affect the system performance. The majority of the information is obtained by constantly monitoring the UNIBUS and Cache/Memory bus transactions. The collected information is fed to the programmable patch panel and microcomputer in the Hardware Monitor.

The signals from most probe points are monitored and a number of pre-calculations are accomplished on a Hardware Monitor board inside the 21(V) system. Many signals are timed and counted and sent to the programmable patchboard in the form of event indicator and a vector of signals associated with specific events. Each event has a specific format of signals associated with it.

The events selected constitute most of the significant computer operations occurring inside the 21(V) system. The probe point for these events are taken from UNIBUS, Memory/Cache, and Front Panel probe points to simplify the measurement technique and the number of wires needed to be plugged into the measured system. In a few cases the signals were not available at standard output locations and wire wrap probes would be used.

3.3.1 Vector Definition

The following intrinsic events have been selected:

- o Fetch Instructions

  The Fetch signal is obtained by a special probe inside the processor or from CPU ROM micro-address observations. The information obtained in the Fetch is recorded in the address data lines being monitored on the UNIBUS. The time in micro-seconds needed for the transaction is recorded also.

- o Memory Read.

  In many cases it is necessary to read data stored in memory. Each time the processor asks for data in storage this event will record the address, data and time necessary for the transaction.

- o Memory Write.

  Same as the memory read above but the data observed is being stored in memory.

- o NPR - NPG.

  Signals used when requesting and being granted a one or two machine cycle data transaction between a master and slave unit will be detected. Included in the measurements taken will be the master and slave unit causing the NPR-NPG and the time necessary for the transaction to be established.

- o BR - BG.

  The daisy chained system of device priority is used when the processor needs to be interrupted for some type of transfer. In most cases an INTR request follows the BR - BG pair.

- o INTR.

  When an interrupt occurs a new program status word is inserted and the processor completes this new process while the PSW of the old routine that was being executed is put in storage. The parameters measured include the new PSW and the times from request to completion as well as between BR and INTR signals.

- o Error.

  When the processor detects an error signal in its operation it will be analyzed in the Hardware Monitor to detect when the error occurred and what peripherals were involved.

- o Trap.

  When a trap occurs, the operator is interested in knowing all the information leading to a trap. It will be available using FIFO memory in the monitor.

Each of the above events will result in an output of the vector generator to the Programmable Patchboard. Depending on the program selected by the operator, any of these events can be accepted for analysis. In each event a number of measurements will be combined to form an event vector. This vector contains measurements that are of interest to that particular event. Table 3-8 specifies the make-up of the vector and the spaces of information allotted to each measurement. From the data received, it can be interpreted what significant unusual occurrences exist in the event. The history of computer events can be inspected since all the vectors are recorded in a FIFO buffer.

Description	Event #	Vector
Fetch Instruction	1.	(DATA)(ADDRESS)(MSYN-SSYN TIME)(ERROR FLAGS)
Memory Read	2.	(DATA)(ADDRESS)(MSYN-SSYN TIME)(MASTER)(PRIORITY)(ERROR FLAGS)
Memory Write	3.	(DATA)(ADDRESS)(MSYN-SSYN TIME)(MASTER)(PRIORITY)(ERROR FLAGS)
NPR-NPG	4.	(ADDRESS)(REQUEST-TO-GRANT TIME)(REQUEST-TO-COMPLETION TIME)(MASTER)(SLAVE)(ERROR FLAGS)
BR-BG (SACK)	5.	(REQUEST-TO-GRANT TIME)(REQUEST-TO-COMPLETION TIME)(MASTER)(SLAVE)(PRIORITY)(ERROR FLAGS)
BR-BG (INTR)	6.	(DATA)(VIRTUAL ADDRESS)(REQUEST-TO-INTR TIME)(REQUEST-TO-COMPLETION TIME)(NEXT PSW)
Error	7.	(DATA)(ADDRESS)(MASTER)(TYPE ERROR)(NEXT EVENT)
Trap	8.	(ADDRESS)(TYPE TRAP)

Table 3-8 Events and Vectors

Each vector exists to simplify the data and group it in a manner that can reduce the workload and complication in reading it. Some signals are measured that occur at a high rate. To keep from interfering with the host computer the concept is to measure and group the signals as close to the probe point as possible. The signals will have some pre-processing done on them to present them to the PP in the proper manner. This pre-processing allows for proper event marking and sequence timing, without over-utilization of PP thus allowing for more efficient use of the PP.

Part of this procedure will allow for five events to be marked when they occur (NPR/NPG, BR 7-4/BG 7-4). At the same time these events are marked, time interval measurements relating to request-to-grant, request-to-completion, grant-to-completion, request-to-grant latency and NPR-NPG contention time are made. These time interval measurements will then be stored in a short term scratch-pad type memory corresponding to the event vector and will be made available to the PP if and when it is requested.

3.3.2 Vector Building Components

There are four basic circuits that will be used to process these events (Figure (3-6). From combinations of them the vector generator will be built.

Circuits I, II will be used to monitor the NPR-NPG, BR-BG events. It will indicate that these events have occurred and will also have the capacity to time the intervals from request to grant and request to the completion of the data transfer. The event will be directly available to the PP and the time intervals will be stored until requested by the PP.

Circuit III will monitor only events pertaining to the BR-BG (7-4) and their grants. It will mark the event (BR-BG) and will also time the intervals of Request-to-Interrupt and Interrupt-to-Completion of the

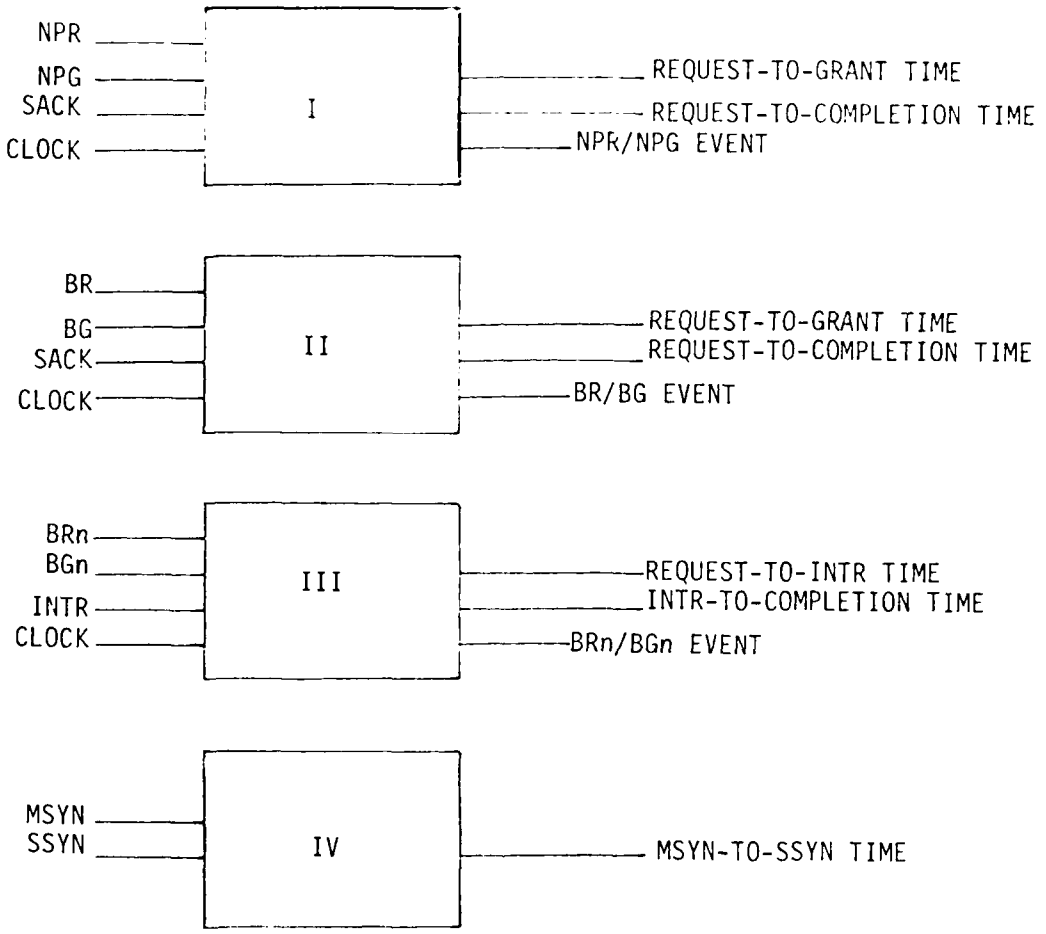


Figure 3-6 Basic Vector Generator Components

data transfer process. The event will be made available directly to the PP and the time intervals will be stored until requested by the PP.

Circuit IV will be used to time MSYN-SSYN.

In addition to these pre-processed results there will be available in the vector the data, address, control lines, as well as mode signals.

Using a latch as a Vector Assembler allows for:

1. Proper collection of vector bits,

2. Latching of vector to ensure correct timing and coordination of vector bits,

3. New vector formation while previous vector is being examined in the PP.

Placing the first vector information into the latch allows for the second vector to be started. This minimizes the delay of the vector transfer due to data transfer interlocking. This also allows for the PP to examine and act upon the previous vector before placing it in long term storage.

By designing a "PP Ready" signal, to indicate to the PP that the vector is ready for transmission, it is insured that the transfer into the PP will be orderly.

This design also allows for future vector size modification and vector tailoring to specific 21(V) systems and particular user measurements with minimal time and cost.

3.3.3 Vector Post-Processing

The basic count of the monitor system is provided by this state vector generator. Each new vector represents either a transfer of data, a transfer of control, or both (e.g., NPR data transfer) and the total number of vectors equals the total number of master/slave bus transactions. The most elementary breakdowns are therefore:

- Total data transfers/total number of vectors
- Number of NPR transfers/total number of vectors
- Number of NPR transfers/total number data transfers
- Number of control transfers/total number of vectors
- Number of NPR transfers/number of control transfers
- Number of BR/BG transfers (i.e., number of interrupts)
- Number of CPU instruction fetches/number of vectors

It is presumed that the monitor always counts the total number of vectors and that the microprocessor is maintaining this information. For each of the above measures, one must count just those vector types of interest. Postprocessing will perform the divisions by the "100% reference" values.

It is most interesting to note that only the event type needs be matched. This is easily accomplished by the programmable patchboards associative logic. The logic identifies the counter to be incremented at each occurrence of the selected event type or types. The counters can be read and cleared when the total vector counter overflows or can themselves cause an interrupt when they overflow. Interrupt rates will be lower than that for the total data transfer cycle counters.

As multiple samples may be obtained over a period of minutes, reports can include absolute counts or ratios averaged over a desirable period of "wall clock" time as well as a measure of statistical fluctuation (deviations from the mean) during the selected period.

3.4 Programmable Patchboard

The function of the programmable patchboard is to digest the many probe results into a reasonable number of events to be accounted for by the monitor processor. As a consequence of this modular approach and of intrinsic hardware limitations, the vector generator will handle and pre-process relatively fast events, the programmable patchboard will process medium speed events (500 ns - 10 n sec) and the monitor processor and memory will process relatively slow events.

The following distinctions can be made in the elementary functions of the programmable patchboard:

- Selection of Events
- Counting of Events

It should be clear that a compression of events has to be made, since it will be unrealistic to be monitoring all possible events all of the time. Compression can take place either by filtering or by outright data reduction. Data reduction is the main function of the Vector Generator, while the Programmable Patchboard mainly performs a filtering function.

Historically, hardware monitors have used logic plugboards to make connections between probes and logic devices, and subsequently between those logic devices and counters in the hardware monitor.

A real-time clock is necessary as a reference either to time the utilization of a device or the total expired time during test. More complicated systems will contain data selectors such as decoders, multiplexers, etc., whose inputs and outputs are also routed via the patchboard.

In commercially available hardware monitors, 500 to 1000 plugs are required, which means that setting-up of the plugboards is not only time-consuming but also complicated. Changes in the monitoring procedure will be hard to incorporate and might be error prone. In order to alleviate those obvious shortcomings of manual plugboards, a dynamically programmable approach will be taken. Not only will this approach result in a more reliable monitor, it will greatly improve set-up and modification times, as well as allow time-sharing of the plug-board hardware between several measurement procedures.

3.4.1 Selection of Events

In order to filter events for only those events that are of interest, the input stream can be compared against preset values or value ranges. It should be clear that sequential comparison will be too slow for the total number of inputs to be accommodated in the system at hand. A recent development in chip technology called PLA (programmable logic arrays) greatly reduces the total number of discrete logic elements required to implement the parallel matching function. A PLA is not only able to detect single or multiple combinations of events, it can also, when provided with a feedback register, detect sequences of events. Basically the AND plane provides an associative function on the inputs followed by a multiple response from the OR plane. A PLA with 16 input variables, 8 output variables and space in the AND plane for 48 combinations can, from those 16 inputs, determine whether or not 8 predefined events have taken place; these eight events can be any logical combination of the 48 predefined ones in the AND plane. Unfortunately, LSI PLA's are only one-time statically programmable; once burned-in they cannot be changed.

Various alternatives exist for the matching function implementation:

- o PLAs, as discussed above, burn-in for a particular set of measurements.
- o PLAs, burned-in for a particular fixed set of measurements, but used in conjunction with RAMs for local changes.
- o Associative memory in conjunction with RAM circuits. The associative memory performs the individual matching, while the RAM performs the detection of combinations of matches.
- o RAMs where the data to be matched is used as an address into RAM, preset with logical ones in the locations that are matches. This solution is cheap and bulky, and can only be realistic if the proper segmentation of data items is accomplished, as well as a hierarchy of sub-match handling.

Mc^2 intends to implement the PLA function, or essentially the parallel comparison function, by a concatenation of an associative memory (AM) and a RAM. The associative memory, in concept, will replace the AND plane, while the RAM will implement the OR plane: this realization will, therefore, be dynamically programmable. In addition, the physical segmentation of AND and OR plane will allow the application of techniques that make associative processing especially attractive. LSI AM chips have been developed, though of relatively small dimensions (e.g., 16x16).

The proper dimensioning of this AM-RAM combination or even a hierarchy will have to come from a detailed analysis of the monitoring requirements. It should also be clear that a software interface will have to be implemented that will obtain the actual coding of the AM-RAM from the user-desired combination of monitored events. In other words, the personality of the AM-RAM combination should be obtained from a functional description, not from an input as an array of bits at the gate level.

3.4.2 Counting of Events

Results from Selection and/or Comparison will be counted and/or sent on to the monitor processor. Counting can be done in either a purely count mode or in a relative rate mode when a "clock" signal is combined with the event in question. The results in the counters will be stored in the monitor processor's memory either periodically or triggered by an impending overflow. This information and the real-time clock value at the moment of dumping will be the basis for subsequent post-analysis by the monitor processor or any other so prescribed machine.

When a histogram is required for the performance distribution of a certain event, use can been made of a RAM in which the content of addresses, created from the respective sub-cases of the event, are incremented by one when the corresponding sub-case occurs. This is known as the distribution mode. One should be careful to make sure that the read-increment-write cycle of the RAM is shorter than the interval between two sub-case occurrences. If that is not the case, a multiple-increment-station and inter-leaved memory configuration should be evaluated. Lastly, the RAM's can be used as buffers in a trace-mode operation, again limited by their write-cycle time.

3.4.3 Programmable Patchboard Realization

A combination of buffers, comparitors and counters (Figure 3-7) will result in a programmable patchboard processor that will be highly effective both from the cost as from the versatility point of view. The Associative Memory (AM) contributes to the selection as well as the match function. Some RAM's operate as part of the AM-RAM PLA implementation, others in a count, time, distribution or trace mode.

The device is programmable by setting up values in the associative memory. The User Interface software (see Section 4.2) is responsible for creating combinatorial tables that are to be used by the monitor processor to "program" the programmable patchboard. Bit positions in this table correspond to the logical AND, OR, XOR, and NOT, operations that are handled by manual switches and connectors in currently available commercial machines. The table would also be used to indicate the disposition of the event (increment counter 23, etc.). The bit-for-bit tables will be loaded from the monitor's microprocessor.

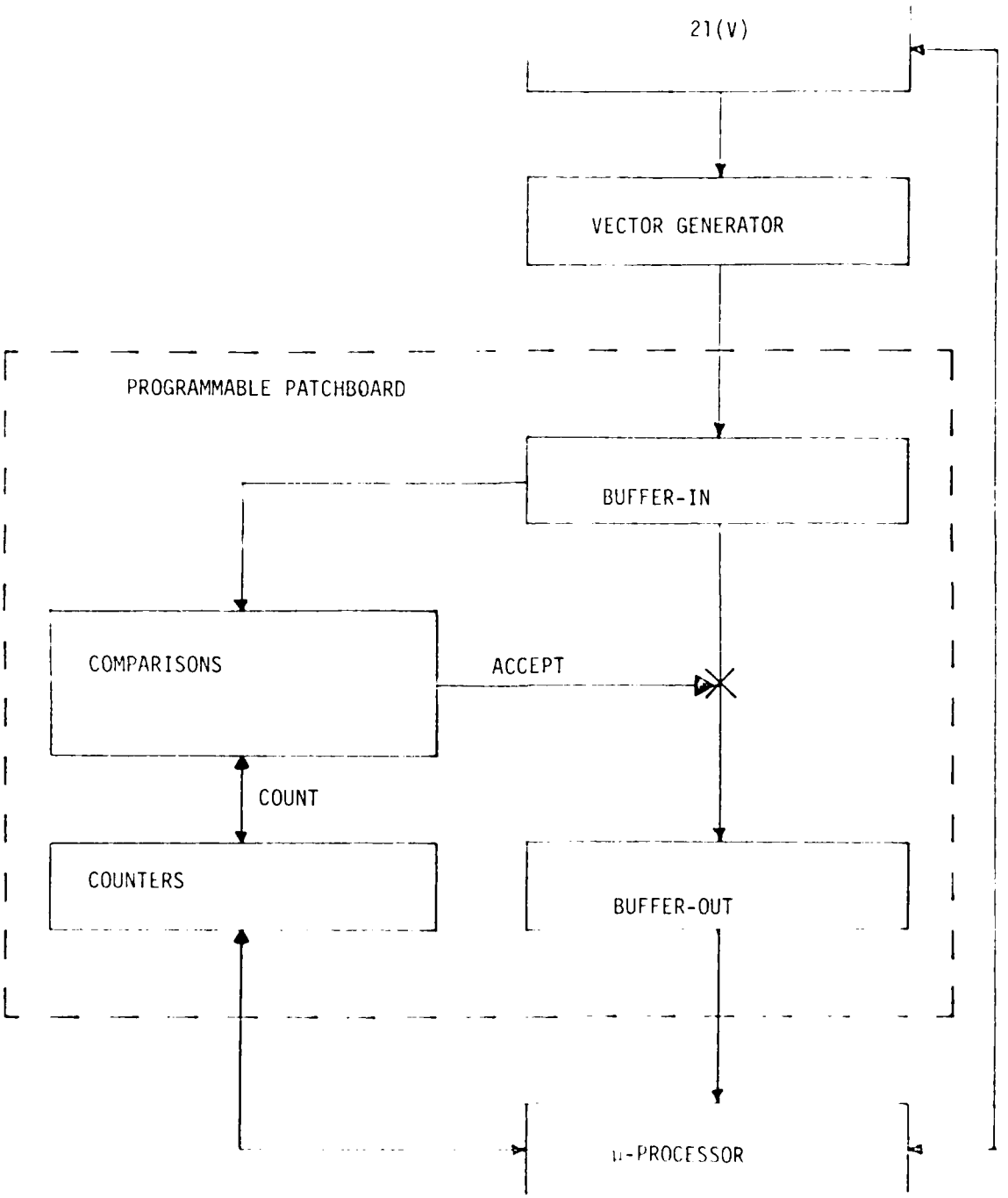


Figure 3-7 Programmable Patchboard

3.5 Monitor Control Unit (MCU)

The functions of the MCU microprocessor are:

- o Monitor Initialization
  - Program load
  - Self test
  - Programmable Patchboard initialization
- o Measurement Data Acquisition
  - Test directive input
  - Data acquisition Control
  - Data path arbitration
  - Data input/storage
- o Data Processing
  - Statistics/correlation
  - Formatting and scaling
  - Abnormal Direction/Fault detection
- o Operator Communication
  - Display selection
  - Data Display/output
  - Status display

The MCU is composed of a standard 16 bit microprocessor, 16K to 32K ROM/RAM, a parallel I/O interface structure for the patchboard, host, and optional peripherals, basic display/control panel, and chassis with integral power supplied.

The microprocessor will be supported with the basic tools required for

easy program development and maintenance. The microprocessor will have the power in terms of instruction set, I/O and interrupt structure, and execution rate of at least a DEC PDP-11/23, TI TMS 9900 or Bunker Ramo IEP. The MCU memory will be a mix of RAM and ROM in variable proportions dependent on the monitor applications.

An optional basic display/control unit will be provided for those configurations not utilizing 21(V) peripherals (Decwriter, LP, etc.) for monitor control and display.

The MCU chassis can provide space and DC power for the programmable patchboard and vector generator logic modules, as well as a means of interconnect (backplane) between these modules. The chassis can accommodate up to 10 cards, and consume no more than 10 1/2 vertical inches of standard 19" RETMA rack space. The chassis will provide self-contained cooling and physical protection for the internal components. The unit will be suitable for table top operation as well as rack mount. The MCU will, subject to space availability, normally be mounted inside the 21(V) in the vicinity (10 ft.) of the CPU cabinet.

With the exception of an AC power and bootstrap/reset switch, there is no absolute requirement for controls to be mounted on the monitor. As the monitor bootstrap, executive, and primary channel interface control software will be resident in ROM, the monitor can be "brought up" except for its measurement definition/control variables, patchboard configuration, and applicable special process software. The latter three items are programmable and are to be resident in monitor RAM. Through the simple provision of a down-line loading facility within ROM, all variable data can be loaded into tne monitor from the host computer. This data has been prepared on the host system prior to testing. Several different tests can thereby be created and stored within the host. The test to be executed can be initialized by a

simple 21(V) console command to load the test by name. Test can be changed without causing any system downtime. Through the power of the host's program development tools, (cross assembler, test definition language interpreter, etc.) test specification can be significantly simplified.

The 21(V) video console and printing peripherals can provide all necessary real time data and "post mortem" reports. As this requires host resources, it may be desirable to attach display and/or report production peripherals directly to the monitor in certain environments. Video display terminals offer the possibility of quiet operation and rapid update for real time information. Hardcopy terminals can double as post mortem printers. Both terminal types offer keyboard input for control at the monitor.

In the following sections we will address the characteristics of the DEC LSI-11/23 and TI TMS 9900 and a current product from Bunker Ramo, the IEP microprocessor. Particular attention will be paid to their respective interrupt handling structures.

3.5.1 Digital Equipment Corporation's LSI-11/23

The LSI-11/23 processor is built around two MOS/LSI chips. The data chip contains all of the logic and arithmetic functions, handles all data and address transfers, external system control and most interchip communication. The control chip contains the Programmable Logic Arrays (PLAS) and sequencing logic for the microprogram.

Communications and control between the data and control chips are handled via a 16-bit micro instruction bus internal to the processor board. This bus also handles communications with the optional memory management unit and floating point unit.

Data transfers between the chips, processor and LSI-Bus are executed on a 16-bit time multiplexed data address line which also handles the interface logic.

Each instruction of the instruction set, which includes both single and double operand instructions, has equal access to I/O interface, and processor registers and memory locations. The 11/23 instruction set is also compatible with existing PDP-11 sets including software, documentation, and associated application notes. An extended instruction set which is standard, allows for hardware integer multiply and divide as well as direct multiple shifting.

All communication between modules, connections with memory and I/O interface elements to the central processor are handled by the LSI-11 Bus. The asynchronous timing of the bus allows each device on the LSI-11 Bus to operate at its own speed, thus promoting better system performance.

Address and data information are multiplexed with data on the bus and interlocking (hand-shaking) is similar to other PDP-11 data transfer routines. The bus provides a fully vectored prioritized interrupt scheme, with nearly an unlimited number of independent interrupts in a system. Each interrupt stimulus has its own vector and service routine assigned to it eliminating the need for device polling. Priority is determined by the electrical closeness of the device, i.e., the closer the device the higher the priority. In addition interrupts may be enabled or disabled for the system or for particular devices through individual control and status registers.

Direct Memory Access Techniques may be utilized to implement faster data transfers for the I/O interface module.

3.5.2 Texas Instruments' TMS 9900

The TMS 9900 is a MOS 16-bit central processor. The instruction set of the 9900 is comparable to any minicomputer and it employs a unique memory to memory architecture with multiple register files that allow for faster response to interrupts and greater programming flexibility. The 9900 is completely compatible with MOS T^2L Logic and memory and is completely supported by Texas Instruments.

The memory architecture is based on a 16-bit (2 x 8 bit bytes) word thus making all memory location even addresses.

Sixteen interrupt levels, 0 through 15, are available on the 9900 with 0, the highest level priority reserved for reset. 1 through 15 are available for external devices and may be shared by several device interrupts. The interrupt code (IC0 through IC3) is compared continuously by the 9900 with the interrupt bit mask contained in status register (ST) bits 12 through 15. When the level of the pending interrupt is less than or equal to the enable mask, the processor recognizes the interrupt and initiates a context switch following completion of the current instruction. The processor fetches the new context Workspace Pointer (WP) and Program Counter (PC) from the interrupt vector locations. Then, the previous context WP, PC, and ST are stored in workspace registers 13, 14, and 15, respectively, of the new workspace. The TMS 9900 then forces the interrupt mask to a value that is one less than the level of the interrupt being serviced, except for level-zero interrupt, which loads zero into the mask. This allows only interrupts of higher priority to interrupt a service routine. The processor also inhibits interrupts until the first instruction of the service routine has been executed to preserve program linkage should a higher priority interrupt occur. All interrupt requests should remain active until recognized by the processor in the device-service routine. The

individual service routines must reset the interrupt requests before the routine is complete.

If a higher priority interrupt occurs, a second context switch occurs to service the higher priority interrupt. When that routine is complete, a return instruction restores the first service routine parameters to the processor to complete processing of the lower-priority interrupt. All interrupt subroutines should terminate with the return instruction to restore original program parameters.

3.5.3 Bunker Ramo's IEP (Information Exchange Processor) Microprocessor

The IEP microprocessor is a powerful, high speed, multi-task CPU, designed for use in modern distributed processing systems. The microprocessor's 16 bit architecture, capable instruction set and ability to address large memories enables it to be used in a wide variety of applications. Yet, the microprocessor is relatively small and inexpensively being assembled on a single printed circuit board along with 8,192 words of high speed dual ported memory.

The IEP microprocessor uses a mixture of hardware and memory registers to achieve a large register set. This combination permits the execution speed of individual programs as well as context switching between programs to be very fast.

Most processors use only one type of general purpose register. The registers are either hardware registers or they are memory words which are addressed as if they were implemented in hardware (memory registers).

Computers which use a memory register structure, such as the TI 990 computer, are able to context switch very quickly. They have this ability because only the program counter and the register set pointer need be saved during an interrupt. The register set pointer defines where the memory registers for the current program are located in memory. Each program running in the computer has its own set of memory registers. The problem with this architecture is that the execution speed of individual programs is sluggish since operands and operand addresses must always be read from memory before an instruction can be executed. Computers which use hardware registers, such as the DEC PDP 11/34 computer, keep the commonly used operands and operand pointers in their hardware registers where they are immediately available for use by the program instructions. As a result, programs execute faster. However, the hardware registers are shared by all programs running in the computer. The contents of the registers must therefore, be saved before the interrupting program can use them. This slows down program context switching.

In contrast, the IEP microprocessor provides each of its program channels with 16 memory registers and also provides 7 hardware registers which all program channels share. Up to 256 program channels may be running concurrently within the microprocessor. The hardware registers temporarily hold the operand pointers, counters, accumulator and shift registers upon which a program is currently operating. The memory registers hold program control words, stack pointers, counters, operand pointers not currently in use, program entry points and intra-IEP mailboxes. During an interrupt, only the first three hardware registers are automatically saved. The remaining four may be saved under program control if they are needed by the interrupting program. The sixteen memory registers associated with the program being interrupted do not need to be saved as they are already in a dedicated portion of memory.

As a result, each program running in the IEP microprocessor has available 23 general purpose registers which it may use. Of these, only the 7 hardware registers used for data and address manipulation need to be saved during an interrupt. This architecture allows the IEP microprocessor to execute programs very fast and to quickly switch context between programs.

The IEP microprocessor executes 80 different types of instructions, the instruction set includes all of the move, arithmetic, logic, shift and jump instructions normally found in mini and microprocessors. Multiply and divide instructions found in some mini and microprocessors have not been included in the IEP instruction set. These functions will be performed by installing a small high-speed floating point processor in the IEP if required.

The IEP instruction set is particularly powerful in the area of program jumps. All IEP compare and test instructions perform an immediate jump if the conditions of the compare or test are satisfied. As a result, the IEP microprocessor performs examine and jump functions in one instruction whereas other processors typical require two instructions for the same task. In addition, the IEP microprocessor performs more types of compare and test instructions than are commonly found in other processors. A heavy emphasis is placed on jump functions since a substantial portion of IEP programs will be control oriented and thus contain a large number of decision loops.

The IEP instruction set fully supports the writing of position independent code. Position independence is particularly important since IEP application programs are typically built by linking together off-the-shelf subroutines. The combination and order in which subroutines are linked continously changes from application to application. As a result the ability to write position

independent subroutines significantly improves the cost effectiveness of IEP systems. Position independent code is supported through the program option of selecting PC relative addresses for all compare, test, jump and intra-page call instructions and by the available operand addressing modes.

Re-entrant program code is also fully supported through the microprocessor's multiple stack, memory register and channel I/O addressing structure. Typically a number of IEP channels are all driving the same type of I/O device or communication circuit. Re-entrant code permits all of these channels to share a single copy of the IEP application program, thereby saving a considerable amount of program memory space.

The interrupt structure, multiple stack and memory register architecture of the IEP microprocessor permits up to 256 concurrent programs to be running within the processor. This multi-task capability is essential to IEP system performance since real time interleaved processing of several I/O channels is the normal intended mode of IEP operation.

3.6 Unibus Interface Card

The Unibus Interface Module provides the monitor control unit's microprocessor with the means of accessing any main memory location, any peripheral device register in the external (UNIBUS) page, and the CPU program accessible registers. Physical access is via a standard UNIBUS interface and cable. By means of the Unibus Interface Module (UIM), the monitor can use the 21(V) UNIBUS facilities to exchange data with and control all of the above devices.

The DR11-B, a Unibus I/O Card, operates directly to or from memory rather than using program-controlled data transfers and is a general purpose DMA interface to the Unibus (Figure 3-8).

The interface consists of four registers and its operations is initiated under program control. When data is requested by the user device, the DR11-B performs a Data-In Transfer (DATI) and loads its data register with information from the referenced address. If a "write" is requested by the user device, a Data-Out Transfer (DATO) is performed moving data from the user devices data registers to the appropriate address. Also a number of control lines allow for burstmodes, byte addressing and read-modify-restore operations.

The UNIBUS Interface provides the means of communication between the host and the microprocessor unit. It allows the results of the software monitoring activities on the host to be transferred to the microprocessor to be correlated with the hardware monitoring results.

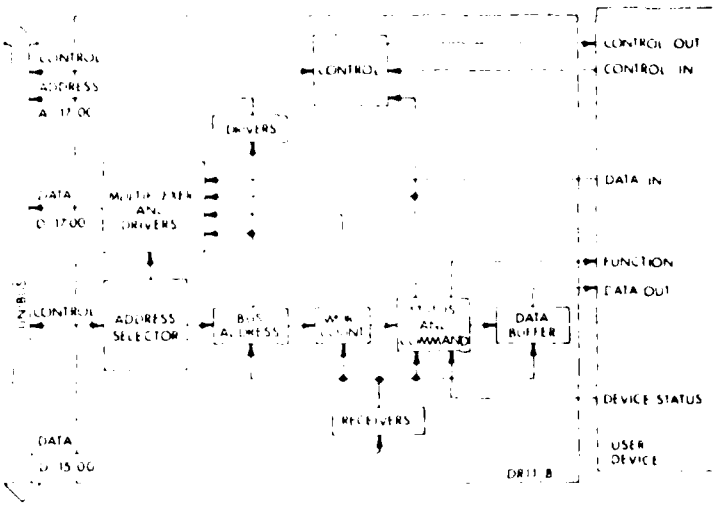


Figure 3-8 DR11-B System

4.0 SOFTWARE FOR HYBRID MONITOR

Software is a critical ingredient in ensuring that the 21(V) monitor can and will meet design objectives in terms of flexibility and simplicity of operation. Flexibility is believed much more than an unspecific ability to handle the wide variety of measurement definitions and combinations made possible by the hardware design. Flexibility must include specific provisions to properly deal with the range of user needs outlined in Section 2.0 and to deal with several levels of user sophistication and knowledge of both measurement techniques and hardware/ software technology.

Monitor software is naturally partitioned by consideration of the machine upon which it executes. The hybrid monitor concept entails cooperative measurement through actions of both the host 21(V) and the monitor's microprocessor controller, and it is necessary to provide software for both machines. To minimize development costs and to maximize monitor measurement bandwidth, much of the most complex software is to be developed for the host itself.

As conceived, there are four unique and distinct functions that can be performed by the monitor - each served by its own set of software:

- o Basic measurement definition and measurement set compilation. Measurement definition and measurement set compilation is analogues to the design coding and task building of a computer program. It is a precise operation that would involve assembly level language programming for the microprocessor and is performed by technically oriented programmer/analysts and engineers in response to evolving requirements for specific measurements and for groups of associated or simultaneous measurements. With the exception of code debugging,

there is no requirement for real time operations and, in any case, there is no need to use the operational target machine which will eventually be measured. Yet the full power and efficiency of a large machine can be utilized to rapidly and efficiently perform this function.

- Selection/qualification/invocation of a measurement set.
  Selection/qualification/invocation is akey user interface. These routines "configure" the monitor by loading/enabling that software to be used for a selected measurement set. This software is used interactively via the operational system's console and only temporarily uses host system resources to provide interactive discourse and actual loading of microprocessor software. The programs need not be resident when they are not being used. The level of interactive discourse is designed for use by computer operator and/or system manager. It is menu-oriented and uses basic English and an easily learned command set.

- Measurement.
  The measurement software is resident in the microprocessor during measurement operations. Since most of the organizational and operator interface work has already been accomplished, the measurement software itself is compact and requires minimal host resources. Depending upon the measurement type and duration (and also upon the level of extra monitor peripherals attached), some host peripheral loading could be encountered in storing measurements data for later (post) processing.

- Reporting.
  Reporting software ranges from the most simplistic microprocessor routines that would drive a near-real-time display, to more complex, time-of-day oriented programs that would reside on the host and utilize the host's printing and disk storage facilities. Although the latter case would introduce additional host loading, the programs could be run at low priority to minimize the impact on other system operations.

4.1 Software Organization

A hybrid monitor will examine both the host computer's hardware and software states. The hardware component of the hybrid monitor will contain the programmable logic units (load modules) specifying logical combinations of events or sequences of events to allow patchboard switching by remote control. The software component of the hybrid monitor is a very small module or modules which reside in the host and which are transparent to the host operating system. This software traps events such as Emulated Trap (EMT) directives and activation of tasks.

Each basic measurement, be it performed by host software intercept techniques, by programmable patchboard/microprocessor setups, or by combinations of these methods is associated with specific software and table structures within the system. The software responds to specific hardware events and the table structures define "programmable" hardware configurations (such as the patchboard) to provide these specific events. Measurement data storage organization for use by measurement and reporting routines is also kept in table form. Standards can be established for software interfaces, table organization, register use, interrupt handling, etc. so that measurement modules consisting of necessary code and structures for a given measurement can be system-

atically developed with assurance that they can be integrated with other modules within a single, uniform, controlled software environment. The combination of individual modules and structures in the standard format is called a "patch". Once created and tested, a patch (consisting of at least one module) may be named, saved and subsequently recalled or linked with other patches and/or modules to build complex measurement sets. Figure 4-1 gives a high-level overview of the manner in which patches are interactifely created (or modified) and saved. Figure 4-2 shows how one or more single patch modules are linked and loaded to create the software for the hardware monitoring activity.

The user, after selecting, for example PTCH23 which consists of modules X, Y and Z, then enters, by use of a simple interactive command language, the run time parameters for the monitor, the events he/she wishes to monitor with additional qualifiers if applicable, and the report format requirements. The interactive hybrid monitor front-end edits the language statements (compiler mode), then also edits the parameters entered based on the patch having been selected. When all user data has been collected, edited, and verified, the hardware parameters are decoded and entered into a matrix which is downloaded to the microprocessor as part of the hardware component of the hybrid monitor. The applicable software parameters are put into acceptable format, then passed to the software component of the hybrid monitor.

The hybrid monitor front-end is envisioned as having several parts: the interactive front-end which accepts and displays data to and from the console of the host computer, a compiler to decode the interactive measurement language, and a link-loader to create the composite software load module and transfer it to the microprocessor to perform the hardware measurement collection activities.

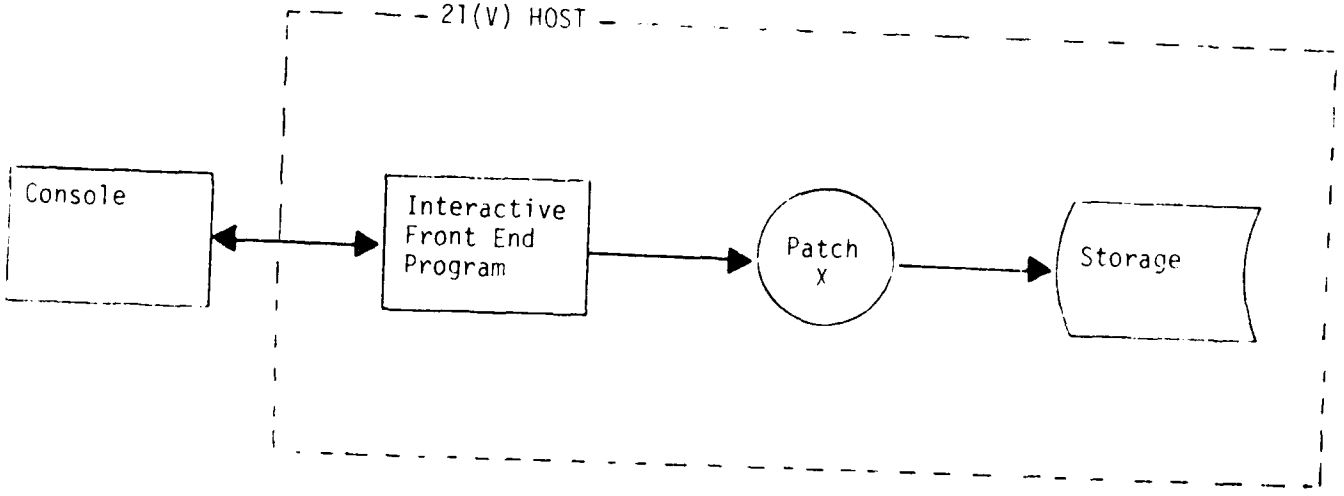


Figure 4-1 Create and Save Patch

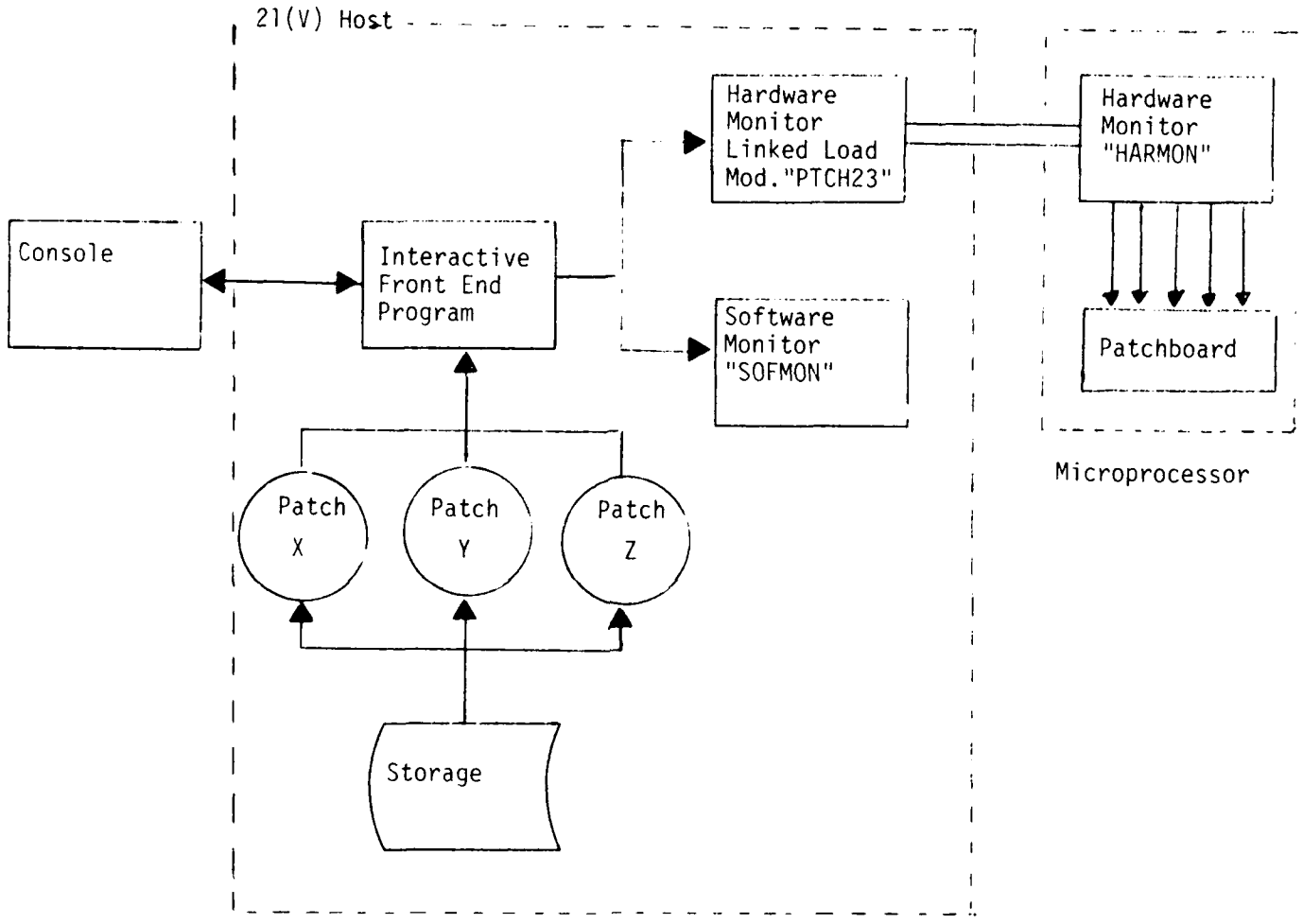


Figure 4-2 Creating Linked Load Module to Initialize and Initiate Monitor

The monitor user command language includes those commands and operations necessary to hierarchically build executable patches from host and microprocessor elements and to store and retrieve such patches by name. Among the capabilities provided by this language is the specification of run time (run-dependent) variables. The host software associated with this feature is activated by console request and/or as a time-scheduled RSX task request. Console requests allow the analyst to invoke execution of a measurement set, change run time parameters, or suspend/cancel a measurement. Run time parameters may be "precanned" so that a clock or activity level related task request could be provided with the desired run time data. With this technique, various measurements can be sequentially performed and/or measurements can be triggered without operator involvement. The invocation of measurement set involves the following steps:

- Create, modify or delete a hardware "soft patch" load module,
- Enter a list of patch modules to be linked into a single patch load module for download to the microprocessor,
- Enter parameters required to initiate and terminate a monitoring session, and to qualify and quantify a selected patchboard configuration,
- Load necessary measurement software and table structures in both the host and the monitor's microprocessor,
- Enter report formats desired as result of the monitoring activity.

The basis for the host measurement software is the fielded Mc^2 software monitor. Whereas Mc^2 believes the proven trap/ATL dispatch intercepts used by the software monitor are the proper method for the

hybrid system, they can neither be left unmodified nor are they, by themselves, sufficient. The major changes involve the introduction of greater modularity and the removal of virtually all in-memory tables and counters. The increased modularity will minimize host core requirements for selected measurement sets. The tables and counters are replaced by equivalent elements within the monitor microprocessor.

The remaining streamlined intercept code now becomes an event detector. Events selected for marking as part of a measurement are first detected by the host resident program. The program then encodes the event and "passes" the code to the external monitor hardware through the monitor's UNIBUS interface. This will be accomplished with significantly less overhead than the software monitor.

The measurement software in the microprocessor accepts the events passed to it by the programmable patchboard. This takes the form of individual events or the accummulated totals of certain events. The microprocessor software prepares these results into the form required by the reporting functions.

The provision of reporting facilities to match the extraordinary flexibility of the measuring device is no simple feat. Without care, the complexity and cost of the reporting software alone could well exceed the combined cost of the other parts of the monitor.

Mc^2 approaches this by recognizing the need for three major report types:

- Fixed statistical reports considered valuable enough to be automatically produced for each run or which are often used as part of standard operating procedures.

- o Parameterized standard report formats such as histogram, Kiviat, and Gantt charts where the user will be able to assign measured variables to the "channels" (axes, strip numbers, etc.) of the report.

- o Structured "core dump" reports.

The fixed report will be tied to specific patches and/or to modules which do not require data storage on the host disk. The format of these reports is chosen to best present the data; graphics could be employed.

The latter two reports are to be organized by "snapshot" - each occurrence of a stored record on disk or by the structured contents of the monitor microprocessor's memory. The basic structure of each snapshot is that of the microprocessor's memory at a given instant so that, given the structure of the memory, individual elements could be selectively extracted for inclusion in the parameterized standard forms.

As it is likely that many patches will involve complex strings of inter-related values, the "core dump" mode is supplied. Major structural boundaries will cause line skips, ejects and/or similar visual cues to aid the analyst in the interpretation of the information.

Special provisions will have to be made when the monitored system in question contains multiple AN/GYQ-21(V)'s. In that case software will be written that will correlate the information from multiple vector generator and programmable patchboard pairs after it has been written into the microprocessor(s) storage.

In the following sections individual attention will be paid to the user interface formats, software monitor functions, microprocessor software and reporting formats. The integration of those software elements constitute the software for the hybrid monitor.

4.2 User Interface

The setup of measurements has to be flexible. An understanding of the term flexible is critical. Each AN/GYQ-21(V) system is uniquely characterized by its hardware configuration, operating system, applications programs, and, most importantly, by the profile of user activity. There is no a priori "obvious measure" to take that is "universal" and independent of the use to which a system is put. For example, systems that generate little I/O are unlikely to have their performance degraded by slower I/O devices or inefficient utilization of these devices. Number crunching problems which would more likely be associated with inefficient code might be detected by program locality measures (average change in program counter between instruction), incidence of page faults or cache "misses", instruction execution rates, timing of innermost program loops, and distribution of instruction types. In a multiprogramming environment the detection of the "offending" routine or subsystem might be made by measuring the amount of CPU time specnt in each task's partition, by determining the relationship of each task's execution times to the system's quantum time slice, by comparing CPU utilization in "overhead" activities such as context switching with CPU utilization in task execution, or by capturing some other "significant event" that typifies a subsystem under study.

The Hybrid Monitor Control Language (HMCL) is a simple measurement definition language which is used to interactively create modules of "patches" and selectively include or modify previously created patches to compose a complete monitor run with the capability to specify run-dependent variables at execution time.

The instruction set allows the user to define events of significance, and specify measurements to be taken when these events occur. The events may range from rather high level actions such as a task becoming active, to machine level events such as signals being passed by a probe. Combinations and sequences of events may be defined as events in their own right - actions occurring simultaneously or consecutively.

Various measurements can be made, triggered by the occurrence of an event. Counts may be kept of the number of instances recognized. Durations and intervals can be measured for events which have beginning and ending points, or for two known successive events. Time stamping may be done upon the occurrence of an event or program trap. Special measurements may be requested of the node pool, UNIBUS or memory (system performance), and in the case of the 11/70, the Fast Bus and Mass bus. Also, reports in the form of histograms of device activities, Kiviat graphs of CPU-I/O activity, or instruction type distributions may be specified.

At the conclusion of each monitoring session, certain statistics are always reported. These include CPU Performance, Floating Point Processor (FPP) performance, and Fault Detection/Isolation.

Additionally, as events, measurements, even whole monitoring runs are defined, they may be manipulated as necessary. Provision is made for storing, retrieving, deleting and listing specified monitor modules, allowing updating or repeating sessions with minimum effort.

Finally, at actual run time, previously defined run-dependent variables can be assigned, giving additional flexibility to the monitoring setup.

Although the exact command invocations and examples shown may be duplicated by a terminal user, they should not be interpreted as representing a necessary sequence of operations. Rather, the examples outline the general function and typical usage of a command or combination of commands. The user should build his/her own language command string to fit the particular monitoring requirement.

The following conventions are used in the description of the syntax of the language. All commands and source text are free form based on correct usage of keywords. Syntax surrounded by square brackets is optional; braces enclosing two objects separated by a slash indicates that one of the two objects must be chosen; keywords are shown in upper case; commas delimiting elements in a list are required; an ellipsis (...) indicates that several objects may be listed.

4.2.1 Event Definition Statements

These statements are used to define events to be considered significant during the run of the monitor. Single events may be combined to create new events. For ease of manipulation during the interactive phase, the events are labeled with a 1-6 character event name.

DEFINE Statement

```
                    ( location = address [THRU address][IF VALUE = value]
                    | SIGNAL = probe #
                    | TASK =   taskname1, taskname2, ...
DEFINE eventname:   < FILE =   filename1, filename2, ...
                    | USER =   userid1, userid2, ...
                    | DIR =    directive1, directive2, ...
```

This statement is used to define events to be recognized by the monitor.

To facilitate measurements at the machine instruction level, two options specifying machine locations or probe signals are provided. Valid locations are:

ILOC - Instruction Location
DLOC - Data Location
PC - Program Counter (Register 7)
Rn - Register n (Registers 0-6)
DEV - Device

Associated with these location identifiers are address variables giving specific values or ranges of values as conditions to be satisfied for the event to occur. For memory locations (ILOC, DLOC, PC, Rn), the addresses consist of eight octal digits, defining an actual memory address. If a peripheral device is specified, the logical unit number,

device address or controller address must also be given with the desired address on the device:

```
DEV = LUN5,256
```

would indicate sector 256 if LUN5 refers to a disk. In addition, the value of the location may be tested by the IF value clause. Any six-digit octal value may be specified.

To specify reception of certain signals to the microprocessor as events, the signal option is provided. By giving the probe number which will transmit the signal, the event is defined to occur whenever a signal is received from the particular probe. If certain combinations or sequences of signals are desired, the component signals may be defined individually and joined by the use of the combine or sequence statements described below.

The TASK, FILE*, and USER options give the capability to specify the occurrences of tasks, file accesses and actions associated with user IDs as events. This gives flexibility at the software level. Valid task names, file names* and user IDs are specified for these options.

The DIR option allows lists of directives to be defined as events. These directives may be combined with other events to give specific statistics or they may be used in their own right to give system measurements. As a large number of directives may exist, a table or user's manual would be provided giving mnemonics or codes for reference.

*File statistics may be monitored dependent on host resources required by the software monitor component to do so (artifact).

COMBINE Statement

COMBINE eventname: event1 {AND/OR} event2 ...

The COMBINE statement provides the capability to define an event as a simultaneous occurrence of other previously defined events. The events may be connected by the key words AND or OR signifying all events must occur or at least one of the events must occur, respectively, for the defined event to be considered as having occurred. Conceivably, with some work, this statement could be extended to allow use of both key words in the same instruction, using parentheses and establishing precedences. This is supported in the current design through use of multiple combine statements defining intermediate events.

SEQUENCE Statement

SEQUENCE *eventname: event1, event2 ...*

The SEQUENCE statement defines an event as the consecutive occurrences of previously defined events. The defined event, "eventname", will have occurred when the events listed are recognized to have occurred in the order in which they appear in this statement.

4.2.2 Measurement Definition Statements

These statements define measurements to be collected when the associated event has occurred.

COUNT Statement

COUNT event

The COUNT statement causes a counter to be maintained representing the number of times "event" has been recognized as having occurred.

TIME Statement

[TOTAL] TIME event

The TIME statement causes the duration of event to be recorded. The optional TOTAL clause indicates the total accumulated duration will be recorded in addition to the individual occurrences. This statistic would be useful for measuring response times or task durations and would have limited application for evaluation at the machine instruction level.

INTERVAL Statement

INTERVAL event1, event2

The INTERVAL statement provides for the measurement of the interval between the occurrence of "event1" and the occurrence of "event2" to be recorded.

TRACE Statement

TRACE event

The TRACE statement causes a time stamp to be saved each time "event" is known to have occurred. This statistic is useful for program debugging.

TRAP Statement

TRAP event

The TRAP statement causes a program trap to occur when the described event is encountered. Processing would proceed as per ordinary program trap. Again, this instruction would usually be used for program debugging or error detection.

4.2.3 Additional Statistics Statement

Inclusion of any of these statements causes a group of statistics to be reported pertaining to the specified resource

PUD Statement

The PUD (Physical Unit Device) statement calls for snapshots to be taken of the Physical Unit Device Table at the beginning and at the end of the monitoring run. The snapshots would record the online and logon status for each device in the table.

NODE Statement

NODE WITH SAMPLE FREQ nnn {SEC/MIN} AND REPORT FREQ nnn {SEC/MIN}

The NODE statement causes activation of the node pool usage monitor at given intervals specified by "nnn" and the appropriate time unit. The node pool usage monitor samples the node pool and reports such statistics as minimums and maximums for nodes held by tasks, nodes allocated and number of nodes each task held at system maximum. Node pool samples are provided at specified intervals and a summary is given at the end of each node pool monitoring period

HIST Statement

HIST dev# WITH PARTS nnnnnn

The HIST statement causes special statistics to be collected to construct histograms of the device activity. The device number indicates which device is to be monitored (a special number would indicate main memory). The value "nnnnnn" indicates the size of the partitions (tracks for disk, address ranges for memory) the device would be separated into to provide sampling intervals for the histogram. This summary would be given at the end of the monitoring period.

SYSTEM Statement

SYSTEM WITH FREQ nnn {SEC/MIN}

The SYSTEM statement causes statistics to be reported pertaining to system performance. Such statistics would be: number of active tasks in memory, memory allocation and task request rate. Again this sampling takes place with a certain frequency specified by "nnn".

UBUS Statement

UBUS

By specifying UBUS, the user will receive UNIBUS performance statistics. These statistics include: UNIBUS Acquisition Time, UNIBUS Occupancy, Interrupt Response Latency, Non-Processor Request (NPR) Latency, Bus Request (BR) Latency, UNIBUS Mapping Register (UMR) Utilization, and Number of Transfers per Second.

FBUS Statement (11/70)

FBUS

With the FBUS statement comes statistics detailing the Fast Bus and cache performance measurements. Reports for these devices would include: Cache Acquisition Latency from CPU, UMAP, and MBC(s), Cache Hit Rates, and Cache/Memory Transfers per Second, and Read and Write Counts. These measurements are available for the PDP 11/70 only.

MBUS Statement (11/70)

MBUS

By specifying MBUS, statistics for the Mass Bus Controllers (MBC) will be obtained. Relevant measurements would be MBC transfers per second (both read and write) and percent busy. If there are several MBCs in the system, the reports would be for each. Again these values may be obtained for the 11/70 only.

ITYPE Statement

ITYPE BY {OPCODE/MODE/REG}[WHEN{PRI=nnn/STATE=state/TASK=taskname[PRI=nnn]}]

The ITYPE statement initiates measurements for an instruction type report. Such a report could be organized by Opcode, Register, or Addressing Mode specified by coding the appropriate word from the first braces. The statistics collected may be restricted to one of the conditions denoted within the second braces. These conditions are:

- STATE - "state" values are KERNEL (Kernel mode), SUPER (supervisor mode) or USER (user mode).

- o TASK - a single "taskname" is specified. USER state is implied. TASK may be further restricted to statistics collection only when it is executing at a specified priority level (PRI=nnn), where "nnn" can range in value from 1 - 250.

- o PRI - priority at system level provides statistics for at total of all tasks executing at the specified priority.

KIVIAT Statement

KIVIAT

The KIVIAT statement calls for a Kiviat Graph to be produced at the end of the monitoring session. At present the graph is envisioned to depict the CPU-I/O interaction for the system as these statistics are readily available (CPU statistics are automatically collected by the monitor and by specification of the UBUS (and FBUS and MBUS for the 11/70) commands(s), statistics will be kept for I/O utilization as well). It is conceivable that other sets of measurements will be found useful to be compared using Kiviat Graph. Parameters would then be required for this statement defining the group of statistics to be used. At the moment, however, the CPU vs. I/O Kiviat Graph appears to be of primary value.

4.2.4 Automatic Statistics

Certain statistics are considered valuable enough to be automatically reported for each monitoring run. They include:

- o CPU Performance, such as: instruction rate, percentage of utilization, percentage of time at priority levels, percentage of time in kernel, supervisor and user modes; also CPU-I/O and CPU/FPP (Floating Point Processor) overlap.
- o FPP Instruction Rate
- o Fault Detection/Isolation counts including: slave faults, error counts, unclaimed NPG, BG counts, memory out of bounds, odd address error, multiple NPR count, multiple BRn counts, glitches on power system, loss of power.

4.2.5 Module Manipulation Statements

These statements give the capability to work with monitor "modules" containing groups of event and measurement statements. This allows the user great flexibility and reduces the work required to create monitoring setups. It also means runs can be set up with a minimum of one command to include a module.

STORE Statement

STORE AS identifier

The STORE command causes the event and measurement definitions made since the previous store command (or start of the session) to be saved for further reference as a module (patch) known as "identifier". This provides for ease and economy of operation as frequently run monitoring setups will be available with the use of a single INCLUDE command. Also implied is the ability to modify a previously run session rather than create an entirely new one.

Some form of direct access storage will have to be maintained for the purpose of saving these modules. The actual method of storage will depend on the implementation chosen.

INCLUDE Statement

INCLUDE identifier

The INCLUDE command retrieves the module specified by "identifier" and makes it available to be linked together with other modules (if required) to form a monitoring run. In addition, once the module is retrieved, it may be further modified as necessary for the current session.

DELETE Statement

DELETE identifier

The DELETE command removes the module specified by identifier from the storage area reserved for the monitor modules.

LIST Statement

LIST [identifier]

The LIST command lists the current event and measurement definitions for the module currently being constructed or as specified by "identifier".

4.2.6 Monitoring Options

These options affect the running of the monitor. The mode output and statistics collection process can be altered as well as setting up test runs to check the performance of the monitor.

SUPER Statement

SUPER

This statement allows the software component of the monitor to be run in supervisor mode. Care must be taken to ensure no other tasks are running in supervisor mode.

STATS Statement

STATS filestring

The STATS statement allows the collection of statistics to be saved at the end of the monitoring run. They will be stored in a file identified by "filestring", a standard RSX file identifier. If this statement is not coded, statistics will be lost after completion of the report phase.

REPORT Statement

REPORT [hh:mm:ss],[{device/NOPRINT}]

This statement causes actions to be taken by the report component other than defaulting to reporting upon completion of the monitoring phase on the system printer. By specifying a time, the report can be delayed to begin at the given time if it is after the completion of the monitoring run. In addition, a device may be assigned as the output unit of the

report or the special option NOPRINT may be coded to suppress the listing of the report. Due to the nature of the histograms and Kiviat Graphs, these special reports must be output to the printer rather than the output unit assigned.

TEST Summary

TEST EVERY hh:mm:ss FROM hh:mm:ss TO hh:mm:ss[;]
[DIR = directive1, directive2, ... [;]]
[NODE = nn[{+ / -}nn [;]]
[DEV = device1, device2, ...]
(Other parameters may be defined).

The TEST statement is used to check the performance of the software monitor. At the given intervals, a test program is invoked causing a known sequence of events to occur. Reports generated by the appropriate monitor setup will then provide a means for determining both the accuracy of the monitor measurements and the artifact caused by the software component of the monitor.

Several options may be used with this statement, to modify events generated by the test program. A semicolon is used to indicate an option follows on the next line. The DIR option lists the directives to be caused by the test program. NODE specifies a number of nodes to be requested, with the capability to raise or lower that number by a fixed increment each invocation of the test program. The DEV option lists the devices events may be directed to. In addition, other parameters may be defined to allow such actions as modifying loop control values of the test program.

It is important to note that the proper monitor event and measurement definitions are used in conjunction with this test facility. Statistics must be collected regarding all events to be compared with the test program, otherwise inconclusive results are to be expected.

4.2.7 Monitor Invocation

This section describes how to start and stop the monitor, and how the special run time assignment facility works.

START at hh:mm:ss FOR hh:mm:ss

This command causes the current event and measurement definitions to be "assembled" and any conflicts to be brought to the attention of the operator for resolution. Such conflicts might be the over-allocation of resources (too many measurements) or that a specified measurement may significantly degrade machine performance. Once the events and measurements and any conflicts are resolved, the hardware monitor load module and qualifying parameters will be passed to the microprocessor for patchboard initialization and resource allocation. The software monitor in the host is initialized for software event detection. At the specified starting time, monitoring will begin and continue for the duration given as the second parameter.

If for some reason it is no longer desired to continue the monitoring run, a STOP command is also provided to terminate execution of the monitor. At any time after the START statement has been entered, the command

STOP

may be entered, causing the monitor to abort. The user will then be prompted for a decision whether or not to report the statistics collected.

An additional feature that should prove worthwhile, especially when using frequently run experiments, is the capability to make assignments to specially defined variables at the time the monitor is invoked. This allows monitoring runs to be designed to measure actions defined

by values not known until run time, for example, to monitor task times. An input variable would be the task name. A special character or sequence of characters would be used at the time of event definition in place of the variable.

In the case of the task name example, the instruction would be coded:

```
DEFINE MYTASK: TASK = ******
```

At run time, since this variable is undefined, the user will be prompted for a value to be assigned. Such a facility greatly increases the usefulness of a monitoring set up.

4.3 Host Software Monitor

The software portion of the hybrid monitor must be kept small to reduce artifact (the side effects of time, memory, and device usage). Although the host monitor, hereafter referred to as SOFMON, is capable of tracking many of the user required measurements, the hardware monitor component, through use of probes, can extract the majority of the same measurements at no cost to system efficiency. These hardware measurements can be meaningless without the interpretive data and control information provided by SOFMON.

The Host Software Monitor will conform to the following design specifications:

- o Transparency
- o Independent maintenance
- o Modularity

The following areas must be considered carefully during SOFMON design:

- o Use of supervisor mode (an option requiring further investigation)
- o Operating system dependency

Figure 4-3 depicts the Host Software Monitor configuration.

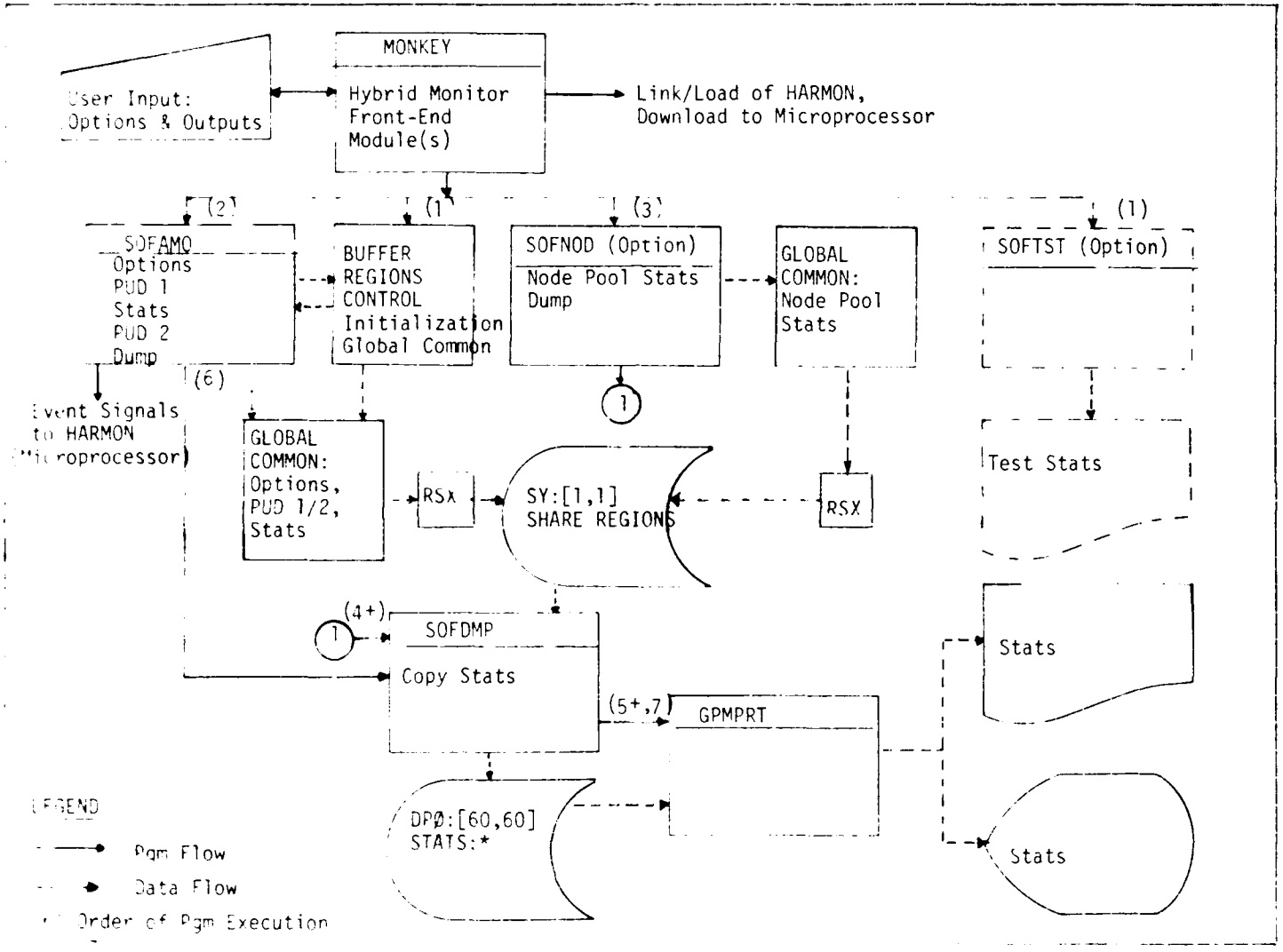


Figure 4-3 Host Software Monitor (SOFMON) Configuration

SOFMON will be completely transparent to both the system user and to the various routines comprising the system and user software. No restrictions or special requirements such as monitor invocations embedded within the system or user programs will be introduced.

Related to the requirement for transparency is the requirement that the monitor not be tied to other software components of the system in such a manner as to force recompilation of one should the other be recompiled or replaced. The is especially important in the case of routines supplied from external sources, such as device handlers.

The design structure of the monitor will be modular so that option modifications can be performed with minimum integration effort, thus affording expansion capability. More importantly, based on option selection, code which is not required for the current monitoring session (example: SOFNOD or SOFTST) will not occupy system memory.

SOFMON artifact can be reduced by dedicating supervisor space to the monitor. This can be done if there are no special tasks such as the Intertask Communications Module (ICM) running in supervisor mode. The interceptive monitor will no longer need to save or restore supervisor registers but can directly address them. The nodes required by the "NODCOD" or kernel portion of SOFMON would be greatly reduced. This option would be implemented interactively in a manner similar to the way the user selects any of the options provided by the interactive

measurement language. The "supervisor mode" option must be selected with care. If another supervisor mode task is executing, selection of this option results in system failure. The default value for the "supervisor mode" option will accommodate other supervisor tasks, at a cost of higher resultant artifact.

The Host Software Monitor, although designed to be general purpose, is dependent on a specific CPU and operating system. Modifications are required to operate under a different operating system (RSX-11D vs. IAS vs. RSX-11M). If the EMT handler address changes or any of the system executive tables are modified, corresponding modifications must be made to monitor code. Care must be taken to thoroughly document monitor code so that changes, if required, can be easily made. The changes are particularly significant between the RSX-11D and RSX-11M operating systems because of changes in the area of inter-task communications. Separate software monitor packages are required for each operating system.

4.3.1 Fundamental Software Monitor Techniques

4.3.1.1 Interceptive Monitor (SOFAMO)

RSX executive software provides a number of services initiated by the Emulated Trap (EMT) instruction. This instruction is issued in conjunction with parameters which define the service requested from the executive. I/O initiation, running of tasks, suspension and resumption of tasks, reading and setting event flags, and task exits are examples of services initiated by the EMT 377 instruction. The EMT is issued in conjunction with a directive parameter block (DPB)

which defines the service requested.

The system return from directive services is implemented through the TRAP instruction. The TRAP service routine then transfers control back to the system by means of the RTI (Return from Interrupt).

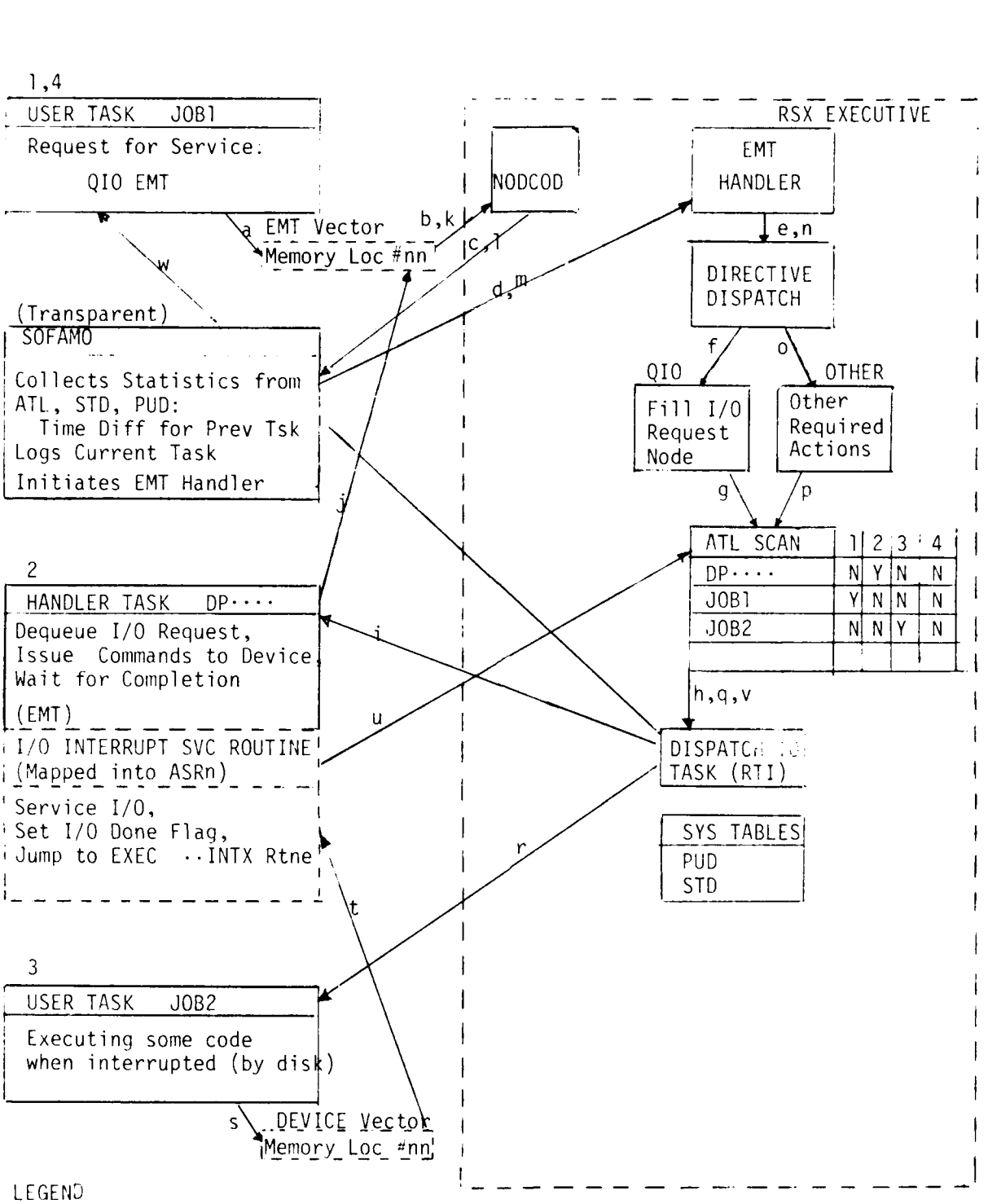
A location in low memory contains the address of the EMT handler. By altering this address to point to the monitoring routines, the EMT can be captured at the start of directive services. The monitor examines the directive parameter block of the EMT, makes appropriate updates to and comparisons with its statistical records, then notifies the hardware monitor component (HARMON) of a significant event or event combination, and passes parameters such as ASR memory bounds for a specific task. SOFAMO then relays the EMT to RSX for normal processing.

Figure 4-4 shows the method by which the interceptive monitor performs its function in the host.

4.3.1.2 Sampling Monitor (SOFNOD)

The Node Pool Monitor is a sampling program with no unusual or special features to distinguish it from the other tasks running in the system. It scans the System Task Directory (STD) on a user-specified time interval, retrieving the node pool usage statistics for each task in the directory if non-zero. Between scans the monitor issues a "Mark Time" directive for the user-specified time interval. If the inter-scan interval is too large, the data collected can be largely meaningless. If the scan window is too small, the system will be unacceptably degraded. The usual method of giving statistics is via a node pool report to console or printer at the specified time interval giving:

- The minimum and maximum number of nodes each task held during the sampling period.



LEGEND

---- User Mode
---- Kernel Mode
--► Flow
Y Executing
N Not Executing
1-4 Order of Execution
a-w Flow Path

Figure 4-4 RSX EMT Handling By The Interceptive Monitor (SOFAMC)

- The minimum and maximum totals of nodes allocated by the system during the sampling period.

- The number of nodes each task held at system maximum.

The results of each sampling are compared to produce the same type of information for the total node pool monitoring period.

Although initiated by the hybrid monitor front-end (MONKEY), SOFNOD is a separate subordinate module whose artifact can be measured. Its run time can be different than that of other hybrid monitoring activities.

Node pool usage statistics are meaningful when monitoring RSX-11D or IAS. RSX-11M design modification in the inter-task communications area reduces the applicability of taking these measurements.

4.3.1.3 Software Monitor Testing

A method for testing and validation should be considered in the design and implementation of SOFMON. Only by careful testing under a controlled environment will accuracy of measurement, and level of monitor artifact be ascertained and documented. A test plan is required to include specification of test programs and the specific conditions under which they run. A test program or programs must exercise every possible EMT directive against each specific device active on the system and keep a count of each directive by type and device unit number or other associated parameters. Times must be documented to include program start and end times, the time each directive was issued and the time the program entered execution again after an EMT interrupt. Testing should also include requests for nodes (blocks of memory) from the node pool and keep a count of nodes used.

A possible approach to testing is the incorporation of a test program within the monitor configuration. An additional option "TEST", would be issued by the user at the terminal. The test module could be run without any monitor functions active or concurrently with any or all of the monitor functions. Reports would be produced both by SOFTST and by the MONRPT portion of the hybrid monitor which could be compared with each other and with other independent test runs. The percentage of artifact caused by implementation of the software monitoring functions or any combination of these functions can be observed and documented. The proposed method of testing allows parametric control of each test. Examples of values that could be entered by the user from his terminal, then passed by MONKEY to SOFTST to modify test conditions are: loop control values to vary the length of an individual test run, the number of times to re-initialize the test on a timed (sampling) basis, or a change in the count of nodes to be requested by the test. Default values would be coded in the test module in the event the user did not want any modifications to the standard test.

4.3.2 Software Monitor (SOFMON) Measurement

Functions listed below can be measured by the software monitor. Within the context of hybrid monitor design, it would in many instances be more cost effective in time and resources for the hardware monitor to take the measurement or measurement group and for the software monitor component to trigger and qualify the action:

AD-A090 026 MEASUREMENT CONCEPT CORP ROME NY F/G 9/2
MINICOMPUTER HARDWARE MONITOR DESIGN.(U)
JUN 80 B MORITZ, H SPAANENBURG, A J LABOUT F30602-79-C-0006
UNCLASSIFIED RADC-TR-80-203 NL

3 OF 3
AD A090026

END
DATE FILMED
11-80
DTIC

- Count of Emulated Trap (EMT) Directives.
  Selection of this option gives a list by EMT name and count for each EMT directives executed during the monitoring period. A sub-option allows a count of all EMTs by name for specified tasks. A sub-option allows the user to input from one to thirty tasks names. The EMT counts are given for each task selected.

- Count of QIO$ EMT Directives by Task.
  Selection of the AIO$ directive option causes the monitor to keep a count for 1-30 specified tasks of the Read Virtual, Read Logical, Write Virtual, Write Logical and other QIO$ directives by device and data transfer quantity.

- Count of Send/Receive EMT Directives by Task Pairs.
  Selection of this option causes SOFMON to record the occurrence of the Send/Receive EMT directives using two counts, one for "Send" type EMTs and one for "Receive" type EMTs. These counts are kept separately for 1 to 39 task pairs with the last send/receive counter and the last receive/send counter reserved for a count of "Others" to notify the user of the system total for the monitoring period.

- Automatic Task Option Selection Statistics.
  If any of the EMT Directive monitoring options have been selected to include task accounting, the following data is collected by task:

  - Time of first activation during monitoring period
  - Time of last exit (task termination) before monitoring terminates

- Number of times task exited during monitoring period
- Number of times task is activated by the Active Task List (ATL) Scan
- Total task CPU time including context switching
- Total RSX-11 service time for EMT being monitored

o Null Task (NULTSK) Statistics.
If any of the Directive options have been selected Null Task statistics are automatically collected and reported by the interceptive monitor NULTSK is only activated by the RSX-11 Executive when no other task is active; therefore NULTSK statistics give a picture of system idle time.

o RSX-11 Service Time.
Similar to the NULTSK statistics, RSX-11 service time can be collected, not only for specified EMTs but for a total of all EMTs, if any of the EMT options have been selected to be monitored.

o Node Pool Usage
The node pool sampling monitor SOFNOD has been described in Section 4.3.1.2. Node pool statistics will be gathered by the separate SOFNOD module executing in the host computer.

o Disk Arm Movement Statistics (Histograms).
Selection of this limited special purpose option causes collection of data to create histograms of disk activity for each of the disk drives on-line during the monitoring activity. The first histogram, in graph form, reflects the frequency with which disk arm movements of a certain distance

(in cylinders) occurred. The second histogram graphically displays, by ranges, the number of times each cylinder was accessed. Monitoring of disk arm movement is hardware dependent. Therefore, monitor code must be revised for each different configuration. Disk arm movement measurements could be taken more effectively by the hardware monitor.

The following monitor control options are available to the user:

- Length of monitor period
- Secondary storage to user-specified UIC, device, and file
- Number of sampling printouts required
- Print device

4.3.3 Extended SOFMON Measurements

Design of the interceptive monitor to include monitoring of the whole range of EMT directives at system or task level permits inclusion of other sub-options with future implementations to capture other specific categories of EMTs with their respective parameters. The configuration of SOFMON also provides an in-depth debugging aid for specific tasks or groups of tasks. SOFMON could be used as a tool to aid in system design and development. Monitor artifact is not an important factor if SOFMON is being used for this purpose. The limiting factor is memory to hold increased statistics. Additional measurement or enhancements of existing measurement functions to be considered follow.

- Dumping Statistics on a Sampling Basis.
  A dump of statistics to secondary storage on a timed or sampling basis provides three benefits:

- The problem of counter overflow as result of an extended monitoring period would be avoided, and

- Data loss due to system failure would be reduced to the statistics gathered since the last dump, and

- More extensive data reduction functions can be performed.

A buffer switching mechanism could be designed to allow the dump processes to proceed while the interceptive monitor continues to increment counts, but using a new set of cleared buffers for data collection while the filled buffers are being dumped. The associated artifact of double-buffering is greater if many options have been selected to be monitored. If the monitor must be suspended for the period of time required to switch buffers, a small quantity of statistics will be lost. Further research may find solutions to these drawbacks.

o System Configuration Changes.
Hardware assignment changes could be intercepted as they occur. Because of additional executable code and memory space required (artifact), two snapshots could be provided - one at the beginning and one at the conclusion of the monitoring session.

o I/O Counts by Filename.
Analysis should be performed to determine the feasibility of capturing I/O counts by specific filename if initiated through File Control Services (FSC). This option could be implemented if the associated artifact is not too great.

4.4 Measurement Collection Software

This section describes the system software architecture for the hardware monitor microprocessor. The microprocessor controls the gathering of information form "Vector Generator" by means of the "programmable patchboard", and supports all communications between the host AN/GYQ-21(V) and the hardware monitor.

Certain features of the proposed architecture are dependent, in their method of implementation, upon the characteristics of the microprocessor chosen. For instance, the LSI-11 compatible microprocessors have but one level of interrupt priority, while the TI9900 series offer fifteen such levels (but fewer interrupt vector locations). It appears that the newest entries in the 16-bit microprocessor fiels (Zilog Z8000, Motorola MC68000, INTEL 8086) have even more versatile interrupt handling capabilities, and may force revisions of this section should they be selected for the monitor implementation.

In its conventional form, the mciroprocessor associated with the hardware monitor will have no peripheral storage or input/output devices, and will depend on the host for its normal operation. Since the microprocessor will contain primarily volatile RAM memory, most microprocessor software modules will reside in storage on the host, and will be sent on an as-needed basis to the microprocessor. The only exception to this depnedence is the software written in non-volatile ROM or PROM storage, which will control the bootstrap sequence and some low-level invariant communications and utility functions (Figure 4-5). As non-volatile memory is logically interchangeable with dynamic memory, it may develop that some of the more frequently-used microprocessor application modules will eventually become permanently resident in the hardware monitor microprocessor.

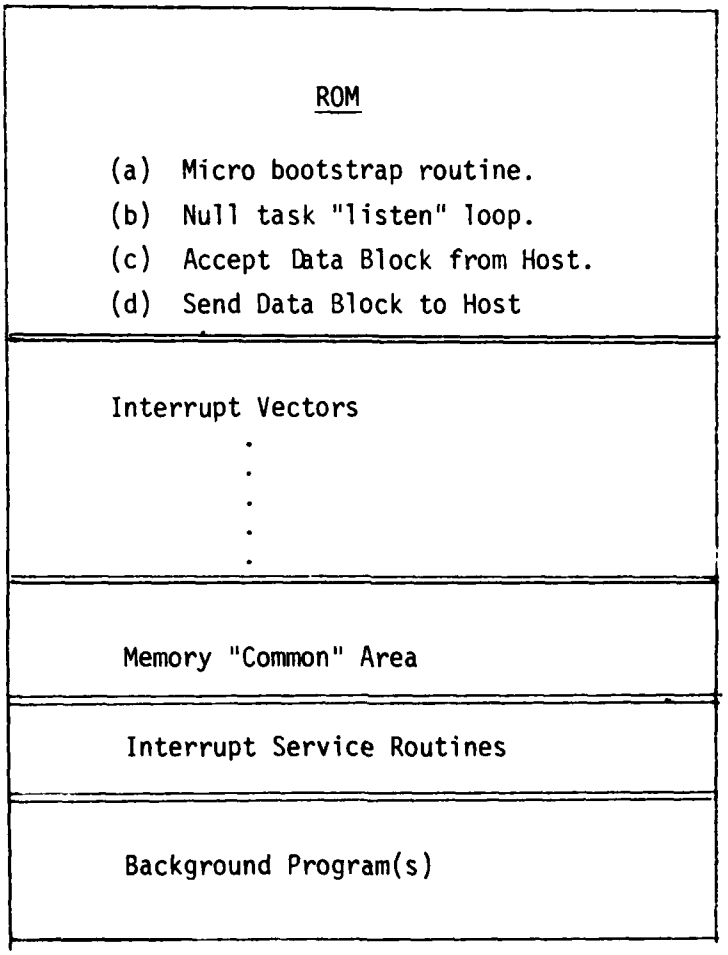


Figure 4-5 Microprocessor Memory Layout

In keeping with the design objective of a relatively low-cost basic monitor, the software architecture of the Hardware Monitor microprocessor is kept quite simple. The actual "operating system" has been spared the clerical responsibility of relocating executable modules and accounting for memory usage: These functions are performed by the test preparation software executing at the host computer. Instead, the microprocessor will behave much as a traffic policeman, directing data flows between the host and the monitor, recognizing service request priorities, and assembling bundles of information for transport to the host-resident monitor report generation software. This relatively simple design has an important benefit - speed. By relocating more complicated processing requirements to the host, the microprocessor will be capable of more simultaneous measurement processes.

The permanent memory of the microprocessor contains routines to "bootstrap" the microprocessor, read data sent from the host and direct it either to microprocessor RAM or the Programmable Patchboard, and write data from the microprocessor to the Host. It is possible that at a later date certain elemental and repetitive measurement functions can be relegated to ROM, but for simplicity, we shall limit discussion to the first three functions.

4.4.1 Bootstrap

The bootstrap procedure is a sequence of hardware and software "events" that resets the microprocessor and peripheral circuitry to a known initial state, and then places the processor in "RUN" mode, executing a set of instructions that "listen" for an interrupt from the host system over the Unibus lines. Because the monitor is not designed to require local I/O

devices, the microprocessor will operate in "slave" mode to the host. The host can then initiate transfer of interrupt vector addresses, interrupt service routines, Programmable Patchboard tables and background programs to the microprocessor, and in the case of background programs, initiate their executions.

The second ROM function is the host-to-microprocessor communications handler. Upon being invoked by UNIBUS interrupt, the micro shall read a 256 byte block from the host. This block may not be completely full, if it contains only a "command" to the micro, or it may contain a command and related data, possibly requiring additional "reads" from the microprocessor's standpoint. In either event, this ROM routine not only accomplishes the interprocessor transfer, but also interprets the function code and modifies to determine where the data is to be loaded (Figure 4-6). Having digested this information, the handler will read subsequent data blocks (where applicable) into the appropriate patchboard or microprocessor locations.

The third ROM function is an analog of the second, in that it controls transfer of data from the micro to the host. The host will always be "expecting" such transfers, and will respond to interrupts generated by the monitor accordingly.

4.4.2 Microprocessor Interrupt Servicing

As the total hardware monitor is basically a passive observer of host system performance, the monitor microprocessor will gather the bulk of its data by responding to interrupts generated by the Programmable Patchboard or interrupts caused by the overflow of counters within

Word 1:	Function Code	Function Code Modifier
Word 2:	Load Address	
Word 3:	Interrupt Vector Address	
Word 4:	Length (Bytes)	
Word 5:	Interrupt Priority	"Next Transaction" Code

Function Code	Function Code Modifier	Function
1		Reboot Microprocessor
2		Accept Data from Host
	1	Load Command Block
	2	Load P.P. Table
	3	Load Interrupt Service Routine
	4	Load Background Task
3		Execute Background Task
4		Terminate Background Task
5		Send Data to Host
	1	Data Dump from Microprocessor
	2	Testing Complete

Figure 4-6 Transfer Command Block

the Programmable Patchboard. This does not mean that microprocessor software will never actively "interrogate" the UNIBUS or Programmable Patchboard, but rather that these activities will be infrequent compared to the servicing of externally generated interrupts.

The interconnection between the physical device (or "pseudo device", where the event being measured is a boolean combination of discrete hardware events) and a particular interrupt request line at the microprocessor, is defined by a table of entries loaded into RAM within the Programmable Patchboard. Logic within the patchboard will use this table to physically connect interrupt sources to the interrupt lines of the microprocessor.

Associated with the patchboard RAM table are the vectored interrupt memory locations within the microprocessor. Each distinct interrupt input line to this processor can initiate cessation of current processor activity, storage of the processor's "current state" variables, and a resumption of execution beginning at a unique memory location. This location, known as an interrupt trap address, contains a branch instruction, to a service routine, unique to that interrupt source, located elsewhere in microprocessor memory.

Each measurement or group of measurements to be performed by the monitor will require downloading of a patchboard interconnection table, a microprocessor interrupt vector table, and appropriate interrupt service routines. The service routines will vary in complexity depending on the expected frequency of the interrupt. For high frequency interrupts, the service routine may do little else than store away data for later processing by a batch reporting routine.

Interrupt service priority presumably will be arbitrated by hardware features of the microprocessor, and should be user-defined. Interrupt structures akin to that of the LSI-11, where priority is, in essence, hardwired, shall hopefully be avoided. It should be possible to define the priority of a service routine as well as an interrupt trap location. (in fact, the TI9900 micro forces one to do this!).

4.4.3 Background Operations

Many monitoring functions will require more than the counting of event occurrences. It may be desirable to associate some measurements with "time stamps", to measure interrelationship among several measurements, or to format results into reports. These less time-critical (read: interruptable) calculations can be run on the microprocessor as a "background" program. The microprocessor would return to the execution of such a program when no interrupts were pending recognition or service.

Background programs would also be down-loaded from the host, with a load point address specified, so that the ROM communication program would know where to "put" the program. By convention, the first location of the program would be a transfer to the starting address of the program.

4.5 Reporting Formats

After the monitor run is completed, and the measurements have been collected, the reporting component of the monitor is activated. Statistics which were requested during the monitor set up are formatted and output to the printer or other specified device. This section gives examples of each HMCL command and an associated sample report.

4.5.1 Event Measurements

Tasks and Devices

Tasks and devices are high-level "events" which merit reporting certain statistics, simply because of their importance to the system. Tasks should always have their number of activations, CPU time and service time reported. As for devices, number of accesses, response time and services times should be given. These statistics will be output whenever a task name or device is defined as an event. If event definitions are input such as:

```
DEFINE EVENTØ: TASK = MYTASK
DEFINE EVENT1: DEV = LUN5
```

then the following statistics would appear on the report:

```
EVENTØ: TASK = MYTASK
52 ACTIVATIONS
0:04.23 sec. CPU
0:00.47 sec SERVICE

EVENT: DEV = LUN5
131 ACCESSES
0:05.43 sec TOTAL ACCESS
0:00.86 sec TOTAL SERVICE
```

Controllers may also be defined as devices by giving the controller address in the event definition. Certain additional measurements pertaining to requests, interrupts and transfers will be reported for each controller defined. An example of a controller being defined is:

```
DEFINE EVENT2:DEV = 774400
```

Where 774400 is the controller address. The statistics reported are:

```
EVENT2:DEV = 774400
742 ACCESSES
677 NPRs
712 INTRs
91804 MEMORY/PERIPHERAL TRANS/SEC.
0:04.87 sec. SERVICE
60.2% BUSY
```

Design of measurement definitions should take into account the statistics automatically reported to eliminate needless duplication of statistics collection.

COUNT Statement

The COUNT Statement causes the number of occurrences of an event to be reported. For example:

```
DEVINE EVENT2: RØ = 177756
COUNT EVENT2
```

keeps track of the number of times Register Ø contained the value 177756. Appearing in the output report is:

```
EVENT2: RØ = 177756
COUNT = 128
```

TIME Statement

Collection of the durations of individual event occurrences is specified by the TIME statement. The TOTAL option, if present, causes the

cumulative event time to be maintained. The statements:

```
DEFINE EVENT4: DIR = QIOW$
TOTAL TIME EVENT4
```

cause monitoring the durations of each QIOW$ directive issued and the total time QIOW$ directives were being executed. Since a large amount of data could be generated with a statement such as this, an acceptable solution is to summarize the durations into a mean and standard deviation of the sampling. The report then shows:

```
EVENT4: DIR = QIOW$
TIME: MEAN = 0.0012 sec., STD.DEV. = .00514, TOTAL = 0.532 sec.
```

INTERVAL Statement

Another measurement requiring summarization is the interval statement to measure the time interval between the occurrence of two events. A situation where this capability is useful might be:

```
DEFINE EVENT1:PC=572
DEFINE EVENT2: PC=576
INTERVAL EVENT1,EVENT2
```

calling for the time intervals between the program counter having the value 572 and then containing the value 576. While this may not occur as often as the QIOW$ directives in the last example, if these instructions appear in a loop, there could be an unwieldy number. Reporting interval times would probably take the form:

```
PC=572, PC=576
INTERVAL: MEAN = 0.0014 sec., STD.DEV. = .00031
```

TRACE Statement

Traces must be handled somewhat differently than the other statistics. What needs to be known here is when the event occurred, why it occurred, and every occurrence of it. Means and standard deviations do not have much significance for traces. Therefore when a trace is specified, every occurrence of the event will cause the time of the occurrence and the values of each component defined in the event to be saved and ultimately reported. Definition statements like:

```
DEFINE EVENT5: DLOC = 10240 THRU 10400
TRACE EVENT5
```

would cause the time and the data location to be output for every time data is accessed between locations 10240 and 10400. The output would look like:

```
EVENT5: DLOC = 10240 THRU 10400
TRACE = 10:25:04.6217, DLOC = 10360
EVENT5: DLOC = 10240 THRU 10400
TRACE = 10:25:06.1825, DLOC = 10244
```

TRACE is a tool which must be carefully exercised to prevent waste of resources.

TRAP Statement

The measurements and reporting for the TRAP statement are much the same as for the TRACE statement. The essential difference is that a program trap caused when the associated event for the trap occurs. An example of the TRAP definition is:

```
DEFINE EVENT6: ILOC = 1132 IF DLOC = Ø
TRAP EVENT6
```

The corresponding report output would be:

```
EVENT6: ILOC = 1132 IF DLOC = Ø
TRAP = 14:52:38.0298, ILOC = 1132, DLOC = Ø
```

Again, the TRAP statement must be used with discretion, otherwise large amounts of output can be expected.

4.5.2 Additional Statistics

This section describes the reporting formats produced by the various additional statistics statements. Each of the statements call for a group of measurements to be collected for the specified resource, therefore, the output of these statistics can be somewhat more structured than the individual event reports. These additional statistics are output after all the event reporting has been completed with the exception of the Physical Unit Device (PUD) report.

PUD Statement

Use of the PUD command causes PUD snapshots to be made at the beginning and ending of the monitor run. The two snapshots are outputted at the beginning and the ending of the monitor report, distinguished by the time of the snapshot. Each device is listed with its on line and logged-on status. The PUD reports are requested with a simple command:

```
PUD
```

Figure 4-7 is an example of a PUD report.

Automatic Statistics Report

Following the individual event statistics will be the additional statistics report. The first group listed will be the automatically

Date: 18 Jan 1979
Time: 12:01:03

PHYSICAL UNIT DEVICE REPORT

PDP-11/45 128K OPERATING UNDER RSX-11D VERSION 6B

Device Name	Unit Number	Online	Logged On
LP	Ø	Yes	
DP	Ø	Yes	
DP	1	No	
MT	Ø	No	
BR	Ø	Yes	
TT	Ø	Yes	Yes
TT	1	Yes	No
TT	2	Yes	Yes

Figure 4-7 Physical Unit Device Report

generated statistics detailing the CPU and FPP performance and the Fault Isolation/Detection measurements. Figure 4-8 is a sample report of these statistics.

NODE Statement

The NODE statement causes the node pool to be sampled with a given frequency, and reports to be produced with a separately specified frequency. The node pool report gives minimum and maximum node usage by task, and a snapshot at system maximum of task node usage, along with the report number, reporting interval and number of samples per report. A NODE statement coded as:

```
NODE WITH SAMPLE FREQ 2 SEC AND REPORT FREQ 50 SEC
```

Calls for node sampling to be done every 2 seconds and reports to be produced every 50 seconds. If the monitoring run goes for one minute report would be output when the statistics are given after the end of the run, summarizing 25 samples. A sample node report is given in Figure 4-9.

SYSTEM Statement

When the SYSTEM statement is specified, statistics regarding the system performance are produced. A sampling frequency is supplied by the user determining when snapshots are taken of three system indicators - number of active tasks in memory, memory allocation, and task request rate. A statement such as:

```
SYSTEM WITH FREQ .5 MIN
```

causes a system snapshot to be taken every half minute. A sample SYSTEM report is given in Figure 4-10.

CPU Performance		Fault Counts	
72.7%	Utilization	3	Slave Faults
26.5%	CPU-I/O Overlap	26	Unclaimed NPG
41.2%	CPU-FPP Overlap	14	Unclaimed BG
5.1%	Supervisor Mode	8	Memory Out of Bounds
13.6%	Kernel Mode	19	Odd Address Error
79.8%	User Mode	7	Multiple NPRs
262675.4	CPU Inst/Sec.	2	Multiple BRs
85729.3	FPP Inst/Sec.	0	Power System Gliches
		0	Losses of Power
		73	Total Error Count
13.5%	Priority Level 8 (NPR)		
44.8%	Priority Level 7 (BR7)		
20.1%	Priority Level 6 (BR6)		
15.2%	Priority Level 5 (BR5)		
6.4%	Priority Level 4 (BR4)		
0%	Priority Level 3		
0%	Priority Level 2		
0%	Priority Level 1		

Figure 4-8 Automatic Statistics Report

```
THIS IS OUTPUT NUMBER  01 OF 01
THE NODE POOL WAS SAMPLED EVERY 002 SECONDS
THE NUMBER OF SAMPLES PER OUTPUT IS 025
          RSX-11 NODE POOL USAGE BY TASK
TASK NAME/MIN/MAX
AMO   003/003  DP....   010/010  LP....   004/004  MONKEY   003/003
NODES   003/005  TT....   063/063  ...MFT   006/006  ...MON   014/014
...RUN   000/003
          SNAPSHOT AT SYSTEM MAX
AMO  /003  DP....  /010  LP....  /004  MONKEY  /003
NODES  /005  TT....  /063  ...MFT  /006  ...MON  /014
...RUN  /003  A  /000
          SYSTEM MIN/MAX...0104/0111
```

Figure 4-9 Node Pool Summary Output

SYSTEM PERFORMANCE STATISTICS

Snapshot Time	Active Task Count	Memory Allocation(%)	Task Request Rate (Per Second)
10:08:00	14	66.1	352.6
10:08:30	19	85.3	308.2
10:09:00	16	72.0	324.7
10:09:30	8	47.6	279.5
10:10:00	13	70.5	296.3

Figure 4-10 System Performance Statistics Report

UBUS Statement

With the UBUS statement comes statistics detaining the UNIBUS performance. A single report is produced at the end of the monitoring run containing various response times, utilizations, and transfer counts. The UBUS statement is entered simply as:

UBUS

A sample UBUS report is Figure 4-11.

FBUS, MBUS Statements (11/70)

The FBUS and MBUS statements call for statistics to be reported on the Fast Bus and Mass Bus, respectively, for the PDP-11/70. The Fast Bus statistics deal with cache latencies and cache/memory tranfer counts, while the Mass Bus statistics report Mass Bus Controller (MBC) utilizations and transfer counts. The FBUS and MBUS reports appear once at the end of the monitoring run. As in the case of UBUS the FBUS and MBUS commands are coded:

FBUS, or
MBUS.

Figure 4-12 and 4-13 give FBUS and MBUS outputs.

KIVIAT Statement

When specified, the KIVIAT statement causes a Kiviat Graph to be output describing the CPU vs. I/O activity of the system. Statistics must be collected during the monitoring run regarding the CPU and I/O activity in order that the chart may be produced. Kiviat Graphs are specified

UNIBUS STATISTICS REPORT

	Mean (Microseconds)	Std. DEV
Unibus Acquisition Time	2.24	0.15
Non-Processor Request Latency	6.77	0.62
Bus Request Latency	3.43	1.07
Interrupt Response Latency	1.84	0.39

Unibus Occupancy 57.3% Unibus Mapping Resister Utilization 42.1%

Transfer Rates (Per Second)

61328 NPR	31578 INTR	14716 Other

Figure 4-11 UNIBUS Statistics Report

FAST BUS STATISTICS REPORT

Cache Acquisition Latencies

	Mean µsec.	Std.DEU
CPU	2.76	0.31
UMAP	3.14	0.26
MBC	2.98	0.42

Cache Hit Rate 71.3%

Transfer Rates (Per Second)

Cache/Memory	Memory/Peripheral
92414 Read	46216 MBC
86320 Write	62024 UNIBUS

Figure 4-12 Fast Bus Statistics Report

MASS BUS STATISTICS REPORT

MBS	% UTILIZATION	TRANSFER RATES (Per Second)
1	48.2%	20562 Read 14608 Write
2	30.9%	13496 Read 9940 Write

Figure 4-13 Mass Bus Statistics Report

by coding:

KIVIAT

The Kiviat Graph pictorially depicts the interaction of several performance components on a circular graph. The resulting shapes can then be interpreted to give a measure of the utilization of the resources involved. One of the most useful Kiviat Graph describes the amount of CPU-I/O overlap present in a system. This graph is constructed by plotting four measurements, for which high utilizations are preferable, on the horizontal and vertical axes, and four "undesirable" measurements, for which low utilizations are preferable, on the diagonal axes. By connecting the plotted points a figure is produced describing the interactions of these measurements. As Figure 4-14 shows the four standard "good" indexes are: CPU active, CPU/IO overlap, any I/O active and program executing in the problem state. The four "bad" quantities are: CPU only, I/O only, CPU not busy, and program executing in the supervisor state.

Figure 4-14 , the so-called "star pattern" represents a system well balanced in terms of continuous and overlapping CPU and channel usage. The maximum utilizations for the horizontal and vertical axes and minimum levels for the diagonal axes contribute to the star-like figure.

The evaluation of Kiviat Graphs is generally not a quantitative process. Experience with the graphs is important to determine the relative efficiency of the system beyond the interpretation of approaching the optimum star shape.

ITYPE Statement

The ITYPE statement gives an instruction type report summarizing

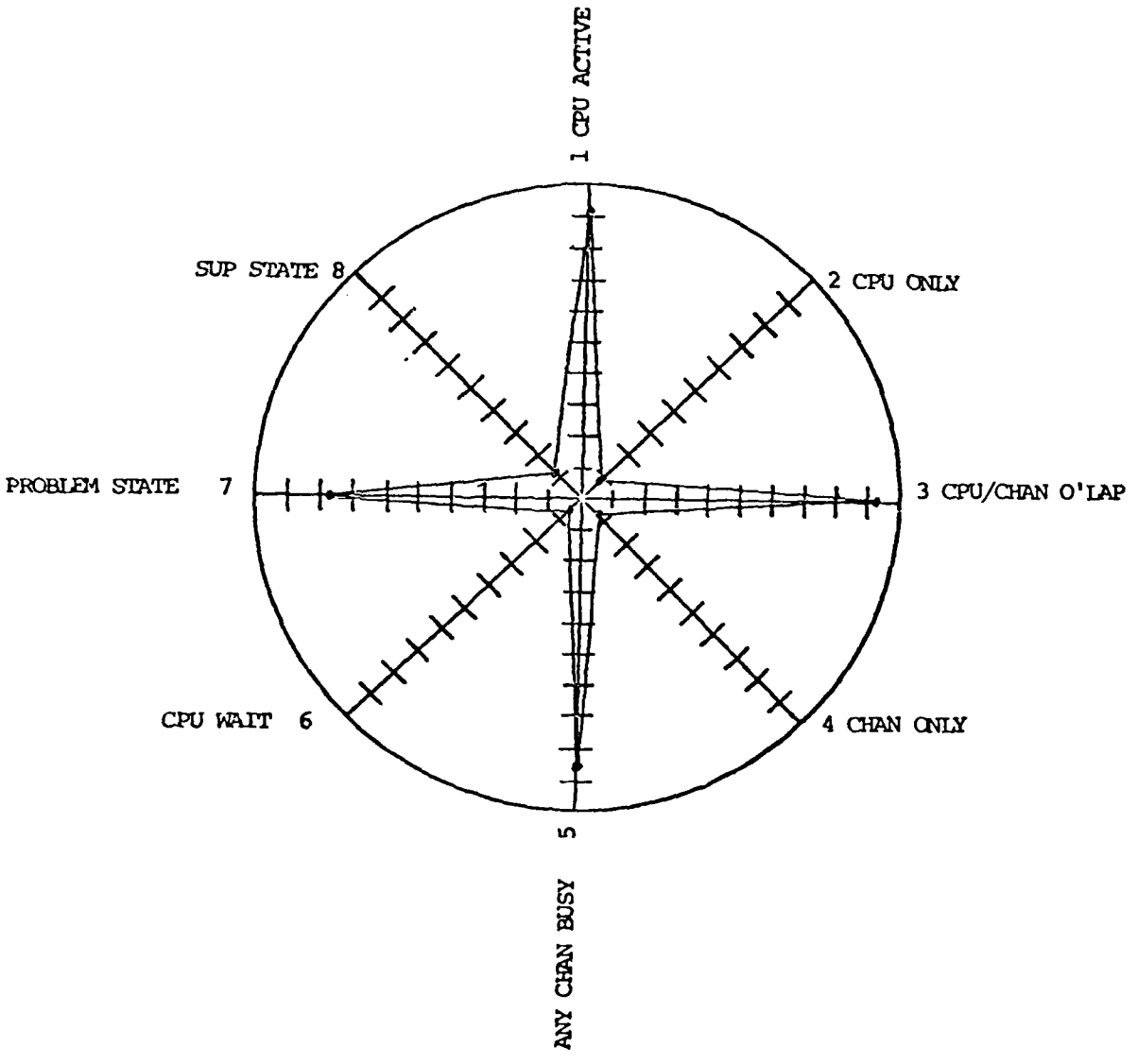


Figure 4-14 Kiviat Graph of Optimum CPU-I/O Utilization

instruction activity during the monitoring run. The instruction summary may be in terms of opcode, addressing mode or registers, optionally restricted to a specified task, priority level or operating state. For each instruction type, the report lists the number executed and the percent of total instructions. An example of an instruction type report specification recording the instruction addressing modes for the task "MYTASK" executing at priority level 75 is:

```
ITYPE BY MODE WHEN TASK = MYTASK, PRI = 75.
```

A sample instruction type report is given in Figure 4-15.

It is important to recognize that the instruction type report can involve a large number of measurements, especially if produced for opcodes. This can have a substantial effect on the number of other measurements collected during the monitoring run.

HIST Statement

The HIST statement produces a histogram for a specified device. Also specified is the size of the partitions (in units appropriate to the device) that the device will be divided into for sampling purposes. It is probable that histograms will be requested for either memory or disk accesses, and the choice will have an effect on the histogram format.

For memory histograms, the memory region is divided into sampling partitions of specified size (perhaps 4096 byte increments), and the count is given of the number of accesses to each partition. Disk histograms, however, involve disk arm movements as well as partitions (specified in number of cylinders) accessed. The disk arm movement histogram reports the number of times the disk arm moves a given

INSTRUCTION TYPE REPORT

BY MODE WHEN TASK = MYTASK, PRI = 75

Mode	Count	% of Total
Register	1528	51.5
Register Deferred	67	2.2
Auto Increment	812	27.4
Auto Increment Deferred	21	.7
Auto Decrement	44	1.5
Auto Increment Deferred	Ø	Ø
Index	392	13.2
Index Deferred	104	3.5
TOTAL	2968	100.0

Figure 4-15 Instruction Type Report

number of cylinders. The disk access histogram is similar to the memory histogram in that it gives number of accesses to each partition. The command:

HIST DPØ WITH PARTS 20

indicates histograms are to be produced for device DPØ (a disk) with each partition containing a range of 20 cylinders. Figures 4-16 and 4-17 give examples of histograms for disk arm movements and cylinder accesses, respectively.

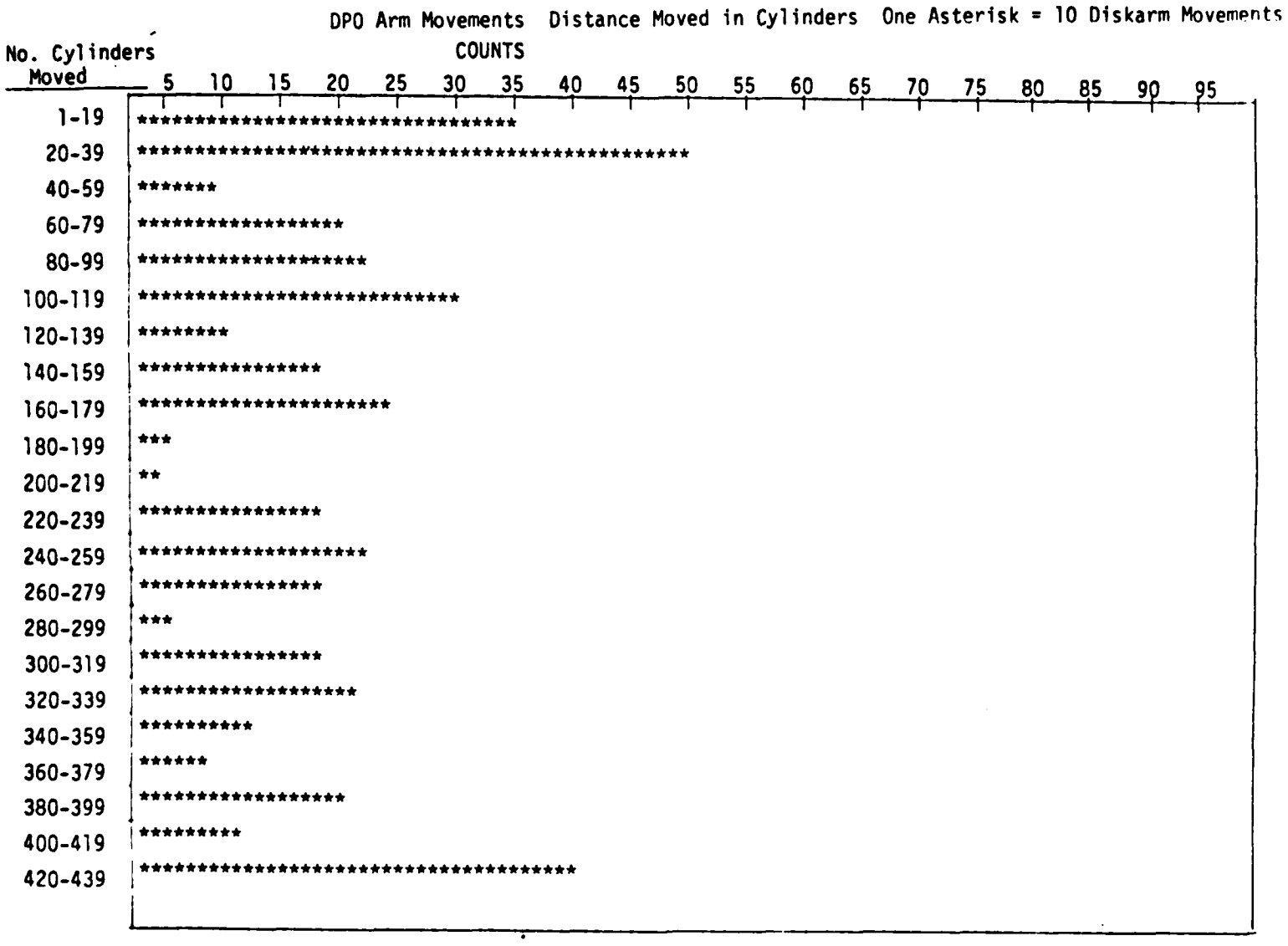


Figure 4-16 Histogram of Disk Arm Movements for DPØ

DPO Counts for Each Cylinder Accessed

One Asterisk = 1 disk count

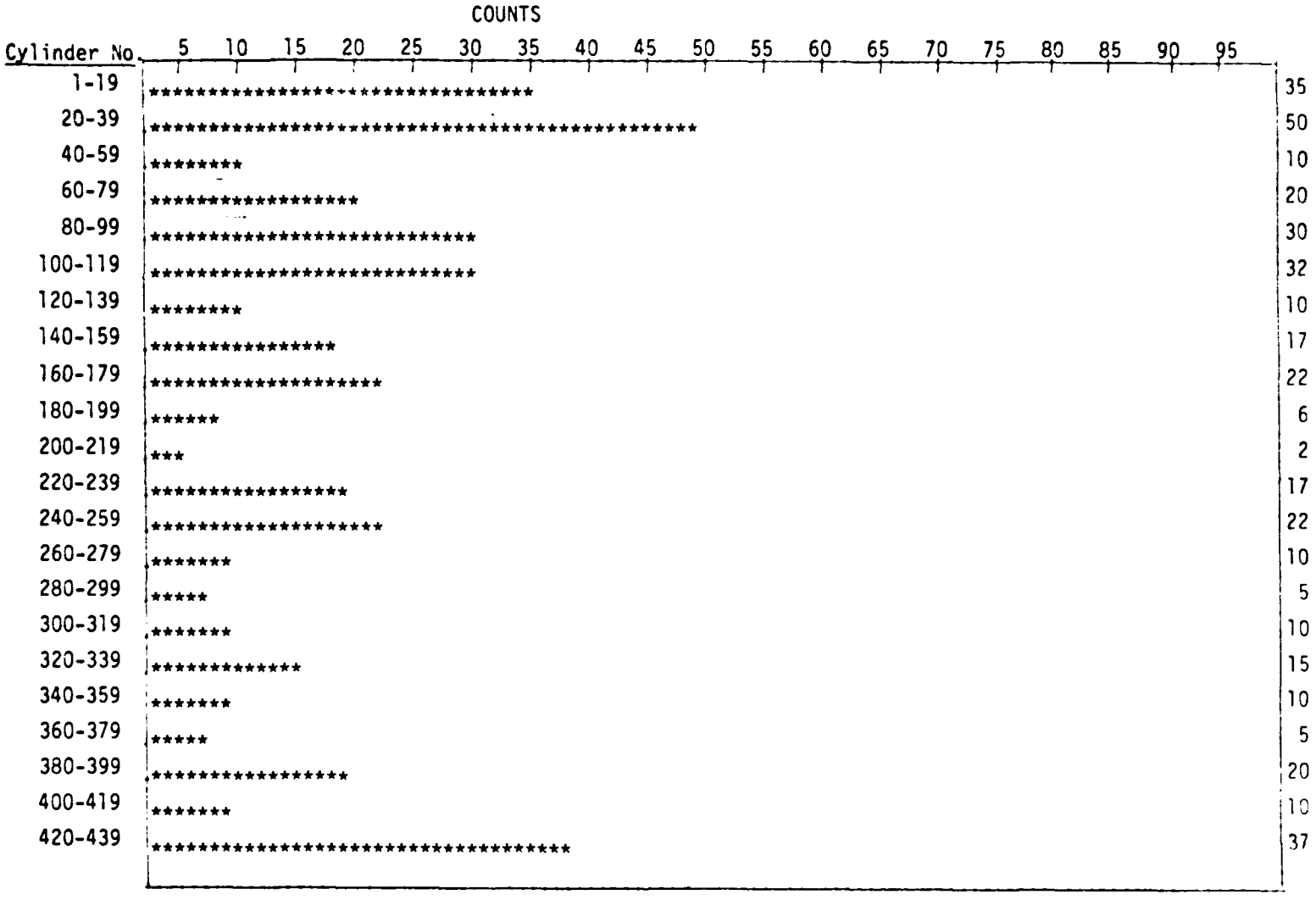


Figure 4-17 Histogram of Counts for Each Cylinder Accessed on DPØ

5.0 SUMMARY

This report has documented the exploratory research that has been conducted by Mc^2 as its contribution to the development of an AN/GYQ-21(V) hardware monitor.

In Section 2.0 a description has been provided of the concept of monitoring, the users involved with monitoring and the monitoring systems that are already in existence. The objective of that section was to put the development of the Mc^2 hybrid monitor into its proper perspective.

Section 3.0 first of all examines the AN/GYQ-21(V) in detail, then describes its interfaces to the monitor. The measurement set considered to be a baseline for performance monitoring is documented in detail as well as the locations of its requisite probe-points. The major contribution of Mc^2 towards the hardware monitor design can be found in the break-up of measurements into three stages: the vector generator, the programmable patchboard and the monitor microprocessor. The eventual hardware specification of those devices awaits the decisions that have to be made concerning the technology involved.

In Section 4.0 specifications for the software involved have been documented. Foremost in the designers' mind has been the ease of user interface with the monitor.

All of the technical and software/organizational techniques presented in the report are within the state-of-the-art. As a result, a microprocessor based performance monitor possessing an extraordinary degree of applicability to all potential users is eminently achievable.

The level of complexity of the modules and components involved in either the vector generator, the programmable patchboard and microprocessor with its accessories do point in the direction of satisfying the production cost requirements of $12,000.

User acceptance has been guaranteed by the lack of user-installed probes and the versatile user interface high level language software.

In short, it is believed that the simplicity of staged measurement handling, maximum utilization of nonsensitive probe interfaces such as the 21(V) busses, and the integration of passive hardware, active hardware, software, and cooperative hybrid measurement/control techniques provide RADC and the end user with a highly useful, low risk product at a cost commensurate with the complexity of the user requirement.

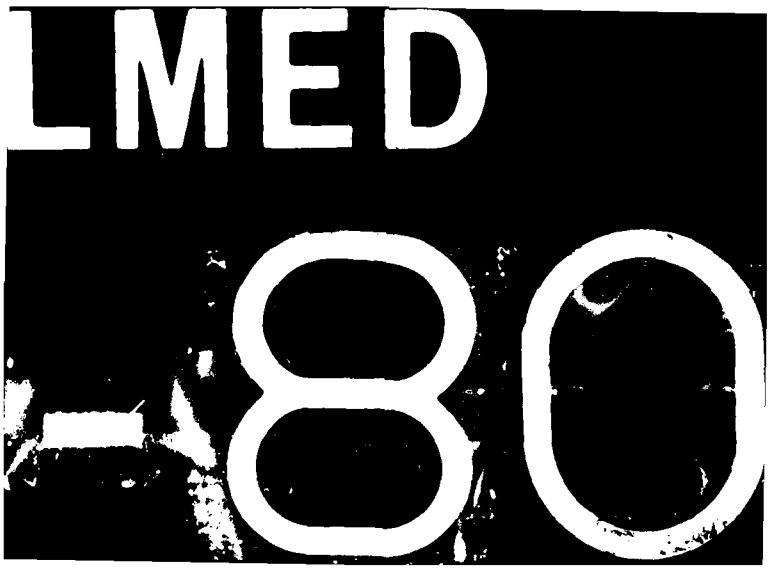
6.0 REFERENCES

Svovodova, L., Computer Performance Measurement and Evaluation Methods: Analysis and Applications
Elsevier, New York 1976

Ferrari, D., Computer Systems Performance Evaluation
Prentice Hall, New Jersey 1978

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.


LMED
80